MoBioTools: A Toolkit to Setup QM/MM Calculations

Tutorials

1 - Reduction Potential: Guanine in Acetonitrile.

2 - Energy Decomposition Analysis: Tyrosine in Water.

3 - Absorption Spectrum: p-Diaminoazobenzene Integrated into the human $Na_v1.4$ Channel.

4 - Emission Spectrum: Luciferine/Luciferase Complex.

---

1 REDUCTION POTENTIAL: GUANINE IN WATER.

In this section, we will determine the one-electron oxidation potential of guanine in water. In order to obtain this property we will run classical molecular dynamics (MD) simulations to explore the conformational space of both the neutral and the cationic version of the system using AMBER. Afterwards, we will use MoBioTools to obtain an ensemble of geometries from each of the trajectories. From the neutral ensemble of geometries we will obtain an average value of the vertical ionization energy (VIE) while the average vertical attachment energy (VAE) will be obtained from the cationic ensemble of configurations. Under the framework of the Marcus theory these two properties can give an approximate value of the reduction free energy of the half-reaction:

$$\Delta G_{red} = \frac{1}{2}\Big(\langle VIE\rangle_R - \langle VAE\rangle_O\Big) - G_e(g) \tag{1}$$

The term $G_e(g) = -0.867$ kcal/mol is the free energy of the electron in the gas phase according to the Fermi-Dirac statistics. Additionally, the one-electron oxidation potential ΔE_{red} can be related to ΔG_{red} by the following equation:

$$\Delta E_{red} = \frac{\Delta G_{red}}{nF} - E^0_{red,SHE} \tag{2}$$

where $E^0_{red,SHE}$ is the absolute one-electron oxidation potential of the reference electrode, which in this case is the standard hydrogen electrode ($E^0_{red,SHE} = 4.28$ V).

1.1 Setup

We will start drawing a guanine molecule in a visualization program. These structures can be taken from any other reference or source. In our case, we have drawn it with the use of the Avogadro package and we have saved the structure as a PDB (see Figure 1). At this point, it is important to highlight a relevant issue. Traditional force fields (FF) are usually parameterized for neutral species. Thus, the use of traditional FFs are not suitable for reproducing the behaviour of a cationic species. In this context, a reparameterization of both the neutral and the cationic species is strongly recommended. In the original paper, we have used the Seminario method for that reparameterization after an optimization of the guanine moiety in both states so that the behaviour of the neutral species is different from the cation. In this tutorial, we will use GAFF2 to model the FF of both species. In addition, we will use ESP charges derived from quantum optimizations of the species to model the electrostatic interactions in a realistic way. However, results will not be accurate at all due to that lack of differentiation in the behaviour of the cation and the neutral molecule.

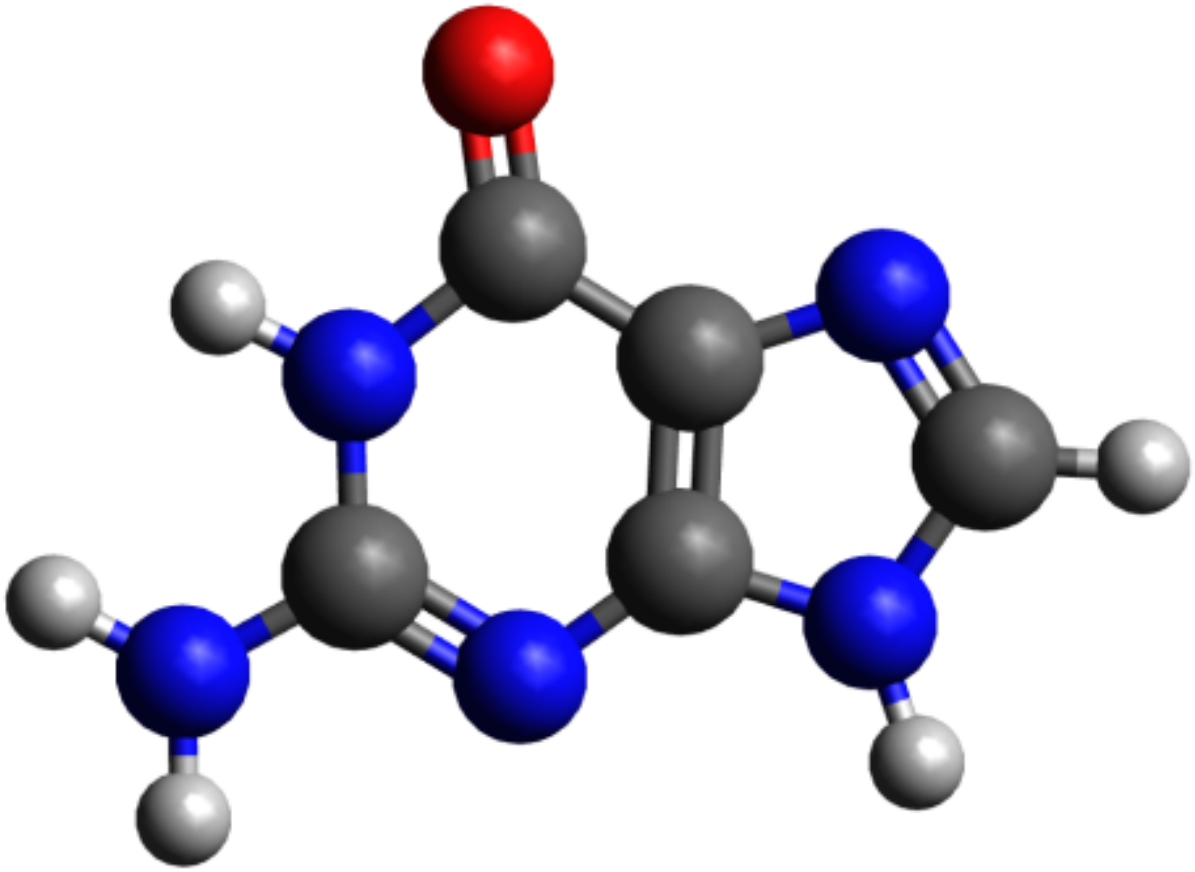


Figure 1: Guanine molecule.

Choose a quantum chemistry package to perform a geometry optimization of guanine in the neutral and cationic state. In our case, we have performed geometry optimizations using the ORCA package at the B3LYP/6-311G(d) level of theory and the CPCM solvent model to account for solvent effects:

```
! Opt B3LYP 6-311G(d) CPCM(Water)
%pal nprocs 8 end
%maxcore 3000
%scf
 maxiter 15000
end
%geom
 maxiter 1500
end
* xyzfile 0 1 guanine.pdb
```

Listing 1: orca_opt.in

Then, we have used the application `orca_2aim` to obtain the wave function of both species:

```
orca_2aim orca_opt
```

And finally, we have used Multiwfn to obatin the ESP charges:

```
orca_opt.wfx
7
18
1
y
0
0
q
```

Listing 2: orca_opt.wfx

With this procedure, we obatined two files with the ESP charges of the neutral and the cationic species. At this point, we will obtain the atom types and the associated parameters from the topology of the PDB of the optimized species. In order to do this, we will use the antechamber application implemented in AMBER typing the following command:

```
antechamber -i filename.pdb -fi pdb -o filename.mol2 -fo mol2 -c bcc -nc 0 -
    m 0

antechamber -i filename.pdb -fi pdb -o filename.mol2 -fo mol2 -c bcc -nc 1 -
    m 1
```

where the first line corresponds to the neutral species and the second line to the cationic molecule. If we now open these mol2 files we will see a column with the calculated charges by antechamber:

We have to replace that column for the one obtained from our ESP calculation at the correspondent species (see Figure 2). Let's now check if there is any parameter left and required in our FF. For that, there exists the `parmchk2` application, which assigns the remaining parameters in terms of their similarity.

```
parmchk2 -i filename.mol2 -o filename.frcmod -f mol2
```

Now, it is the time to build our solvated system. For that purpose, we will use the tleap application also implemented in AMBER. First, generate an input (`filename.in`) like the following:

```
source leaprc.gaff2
source leaprc.water.tip3p
loadamberparams filename.frcmod
UNL = loadmol2 filename.mol2
solvateoct UNL TIP3PBOX 12
(addions2 UNL Cl- 0)
saveamberparm UNL filename.prmtop filename.rst7
quit
```

Listing 3: filename.in

and type the following command:

```
tleap -i filename.in
```

In the input, the line in parenthesis is only required in the cationic system since it is important to neutralize the positive charge of guanine. The generated files (`filename.prmtop` and

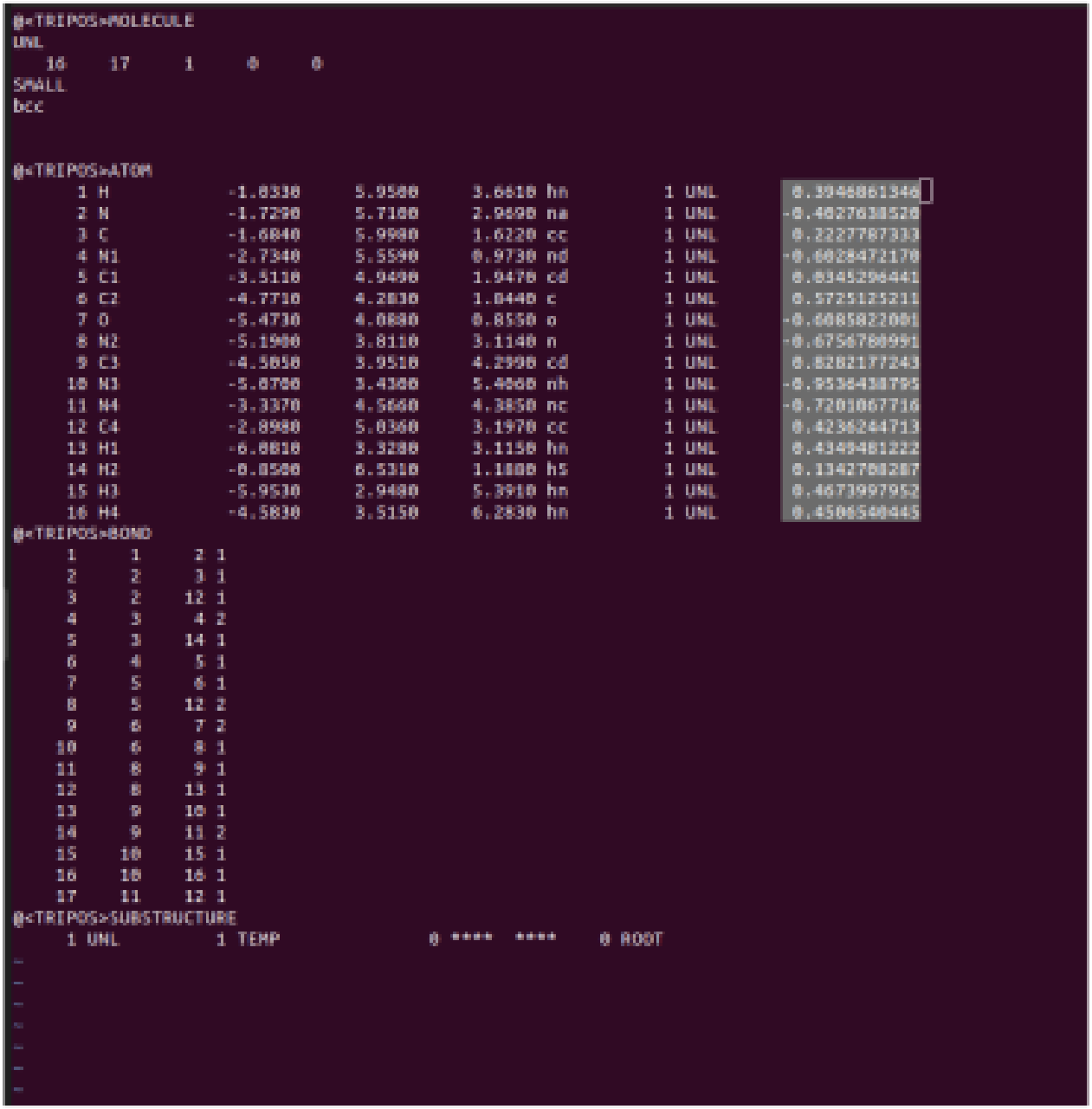


Figure 2: mol2 file where the highligthed region correspond to the charges that must be substituted by the ones calculated by a quantum chemistry package.

filename.rst7) are the parameters and the coordinates files which will be given to AMBER in order to run the classical molecular dynamics situation. Note that this process must be done for the neutral and the cationic species so that two different simulations can be performed, each of them in the correspondent conformational space.

1.2 Molecular Dynamics

At this point, we are able to perform the MD simulations. The sampling procedure will be divided in four different stages: minimization, heating, pressure equilibration and production. For the minimization, the following input (01_Min.in) will be given to AMBER:

```
    Minimization input file in explicit solvent
 &cntrl
    ! Minimization options
    imin=1,         ! Turn on minimization
    ntx=1,
    irest=0,
    maxcyc=10000,   ! Maximum number of minimization cycles
    ncyc=5000,      ! Number of steps of Steepest descent algorithm
    ntr=0,
    ntb=1,          ! Periodic boundary conditions

    ! Potential energy function options
    cut=12.0,       ! Cutoff
    fswitch=10.0,   ! Switching function

    ! Control how often information is printed to the output file
    ntpr=100,       ! Print energies every 100 steps
    ntwx=0,
 /
```

Listing 4: 01_Min.in

In order to perform the minimization, type the following command:

```
pmemd.cuda_SPFP -O -i 01_Min.in -o 01_Min.out -p filename.prmtop -c filename
   .rst7 -r 01_Min.ncrst -ref filename.rst7 -inf 01_Min.mdinfo &
```

Once the minimization is done, let's perform the heating using the following input (02_Heat.in):

```
Heat
 &cntrl
  imin=0,           ! Not a minimization
  ntx=1,
  irest=0,
  nstlim=500000,    ! Number of steps
  dt=0.002,         ! Timestep
  ntf=2,            ! SHAKE algorithm
  ntc=2,            ! SHAKE algorithm
  tempi=0.0,        ! Initial Temperature
  temp0=300.0,      ! Final Temperature
```

```
  ntpr=100,          ! Frequency of printing
  ntwx=100,          ! Fequency of snapshot
  cut=12.0,          ! Cutoff
  fswitch=10.0,      ! Switching function
  ntb=1,             ! Periodic boundary conditions
  ntp=0,             ! No pressure control
  ntr=0,
  ntt=3,             ! Langevin thermostat
  gamma_ln=2.0,      ! Collision frequency
  nmropt=1,          ! Restraints
  ig=-1,             ! Random seed
 /
&wt type='TEMP0', istep1=0, istep2=250000, value1=0.0, value2=300.0 /
&wt type='TEMP0', istep1=250001, istep2=500000, value1=300.0, value2=300.0 /
&wt type='END' /
```

Listing 5: 02_Heat.in

and type:

```
pmemd.cuda_SPFP -O -i 02_Heat.in -o 02_Heat.out -p opt.prmtop -c 01_Min.
    ncrst -r 02_Heat.ncrst -ref 01_Min.ncrst -x 02_Heat.nc -inf 02_Heat.
    mdinfo &}
```

For the pressure equilibration, we will use the model input

```
Equil
 &cntrl
  imin=0,
  ntx=7,
  irest=1,
  nstlim=500000,
  dt=0.002,
  ntf=2,
  ntc=2,
  temp0=300.0,
  ntpr=50000,
  ntwx=50000,
  ntwr=50000,
  cut=12.0,
  fswitch=10.0,
  ntb=2,
  ntp=1,             ! Pressure control (Berendsen barostat)
  ntr=0,
  ntt=3,
  gamma_ln=2.0,
  pres0=1.0,         ! Pressure 1 bar
  comp=44.6,         ! Compressibility coefficient
  taup=2.0,          ! Pressure relaxation time
  ig=-1,
```

```
 /
```

Listing 6: 03_Equil.in

and type:

```
pmemd.cuda_SPFP -O -i 03_Equil.in -o 03_Equil.out -p opt.prmtop -c 02_Heat.
   ncrst -r 03_Equil.ncrst -ref 02_Heat.ncrst -x 03_Equil.nc -inf 03_Equil.
   mdinfo &}
```

Finally, for the production stage we will use the input displayed below

```
Prod
 &cntrl
  imin=0,
  ntx=7,
  irest=1,
  nstlim=100000000,
  dt=0.002,
  ntf=2,
  ntc=2,
  temp0=300.0,
  ntpr=20000,
  ntwx=20000,
  ntwr=20000,
  cut=12.0,
  fswitch=10.0,
  ntb=2,
  ntp=1,             ! Pressure control (Berendsen barostat)
  ntr=0,
  ntt=3,
  gamma_ln=2.0,
  pres0=1.0,         ! Pressure 1 bar
  comp=44.6,         ! Compressibility coefficient
  taup=2.0,          ! Pressure relaxation time
  ig=-1,
 /
```

Listing 7: 04_Prod.in

and type:

```
pmemd.cuda_SPFP -O -i 04_Prod.in -o 04_Prod.out -p opt.prmtop -c 03_Equil.
   ncrst -r 04_Prod.ncrst -ref 03_Equil.ncrst -x 04_Prod.nc -inf 04_Prod.
   mdinfo &
```

1.3 MoBioTools

The `04_Prod.nc` file contains the production trajectory. The full MD protocol must be done twice, one for the neutral guanine and another one for the cation parameterization. Those two `04_Prod.nc` files will be the files that we will use to obtain the ensemble of geometries of the neutral and the cationic species using MoBioTools.

```
&main
tpl = orca
top = filename.prmtop
traj = 04_Prod.nc
qmmask = :UNL
geoms = variabless
solvmask =:WAT
&end
```

Listing 8: orca.main.in

```
&header
&end

&route
!B3LYP/6-311G*
&end
&externchg
&end
&chgspin
0 1 (1 2)
&end
```

Listing 9: orca.tpl.in

If many snapshots are needed, you can use the following script:

```
#!/bin/bash
for number in ` seq 200 20 5000` ; do
   sed -i "s/variabless/$number/g" orca.main.in
   main_qminputs.py -i orca.main.in -t orca.tpl.inp
   sed -i "s/$number/variabless/g" orca.main.in
done
```

Listing 10: more_geoms.sh

1.4 One-electron Oxidation Potential

Now, it is the time to run all the ORCA inputs generated for both, the neutral and the cationic species. For each trajectory, two sets of calculation must be performed: one for the neutral species and one for the cation. In this way, four sets of calculations must be conducted: neutral species from the neutral trajectory, cationic species from the neutral trajectory, neutral species from the cationic trajectory and cationic species from the cationic trajectory. Once these calculations are finished, the only thing required is to take the electronic energies of each geometry and use Eq. 1 to obtain the one-electron oxidation potential of guanine. In out case, this procotol (with Seminario reparameterization) revealed a one-electron oxidation potential of 1.28 ± 0.65 V.

2 ENERGY DECOMPOSITION ANALYSIS: TYROSINE IN WATER.

This section aims at illustrating the ability of the MoBioTools toolkit to easily process MD trajectories into input files for the calculation of interaction energies and their different quantum mechanical energy contributions for a system consisting of two fragments, at the QM/MM level of theory. The chosen test system consists of the aminoacid tyrosine in a water solution, which undergoes configurational sampling by means of classical MD, followed by the Energy Decomposition Analysis (EDA) of the resulting geometry ensemble. An electron density based QM/MM-EDA scheme[1, 2, 3] is used for such purpose, allowing the decomposition of the total interaction energy in its electrostatic, polarization (induction and dispersion), and Pauli or exchange-repulsion components.

2.1 Setup

The geometry file of tyrosine (`tyr.xyz`) is taken from reference4. This file has to be transformed into `.pdb` format. In order to do that, we open the file with `PyMOL` and click on File > Export Molecule > Select tyr > Save

There are some changes that have to be performed in the obtained `tyr.pdb` file:

1. First column: Change the record type HETATM by ATOM by typing
   `sed -i 's/HETATM/ATOM /g' tyr.pdb`
2. Third column: Change atom name from N, C, C, ... to N1, C2, C3, ...
3. Fourth column: When the number is added to the atom name, the column is moved one placed, so it is necessary to delete one space.
4. Fifth column: Change to 1 because there is only one residue.

Then, we generate `tyr.mol2` file using `antechamber` module from `AmberTools20` by writing

```
antechamber -i tyr.pdb -fi pdb -o tyr.mol2 -fo mol2 -c bcc -at gaff -nc 0
```

where the options `-i`, `-fi`, `-o`, `-fo`, `-c`, `-at` and `-nc` correspond to the input name, input format, output name, output format, method to determine the charges, atom type, and the total charge of the system, respectively.

Although this `tyr.mol2` file contains charges, we calculate Merz-Singh-Kollman (MK) charges with B3LYP functional and 6-311g(d) basis set using Gaussian09. When this calculation has finished, we copy MK charges to the `tyr.mol2` file (last column).

The next step is to check if there is any missing parameter using `parmchk2` module from `AmberTools20` by this command

```
parmchk2 -i tyr.mol2 -o tyr.frcmod -f mol2
```

Where the options `-i`, `-o`, and `-f` correspond to the input file, the output file and the input format, respectively.

Then, we generate the library file with `tleap` module from AmberTools20 by typing

```
tleap -f tleap.inp
```

```
source leaprc.gaff2          # source leaprc file for gaff
source leaprc.water.tip3p    # source leaprc file for tip3p
loadamberparams tyr.frcmod   # load tyr parameters
UNK=loadmol2 tyr.mol2        # load whole system
saveoff UNK tyr.lib          # save library of tyr
quit
```

Listing 11: tleap.inp

And finally, we solvate the system using again *tleap* module with the same command as before

```
tleap -f tleap2.inp
```

```
source leaprc.gaff2                                  # source leaprc file
                                                       for gaff
source leaprc.water.tip3p                            # source leaprc file
                                                       for tip3p
loadamberparams tyr.frcmod                           # load tyr parameters
loadoff tyr.lib                                      # load tyr library
TYR=loadpdb tyr.pdb                                  # load whole system
solvateOct TYR TIP3PBOX 12.0                         # solvate system
                                                       with water with
                                                       truncated octahedron
                                                       box of 12.0 A
addions2 TYR Na+ 0                                   # add ions
addions2 TYR Cl- 0                                   # add ions
savepdb TYR tyr_solv.pdb                             # save system
saveamberparm TYR tyr_solv.prmtop tyr_solv.rst7      # save parameters
quit
```

Listing 12: tleap2.inp

2.2 Molecular Dynamics

Once all inputs are prepared, the next step is to run molecular dynamic (MD) simulations. The following amber inputs are used

Minimization

```
&cntrl
   ! Minimization options
   imin=1,          ! Turn on minimization
   ntx=1,           ! Read coordinates  but not velocities
                    ! from ASCII formatted rst7
   irest=0,         ! Do not restart simulation.
   maxcyc=10000,    ! Maximum number of minimization cycles
```

```
   ncyc=5000,      ! 5000 steepest-descent steps, better
                   ! for strained systems
   ntb=1,          ! Periodic boundaries for constant volume
   iwrap=1,        ! Wrap coordinates into primary box

   ! Potential energy function options
   cut=12.0,
   fswitch=10.0,

   ! Control how often information is printed to the output file
   ntpr=100,       ! Print energies every 100 steps
   ntwx=0,         ! No amber mdcrd trajectory file written
/
```

Listing 13: 1_Min.in

Heating

```
&cntrl
   imin=0,         ! No minimization
   irest=0,        ! This is NOT a restart of an old MD simulation
   ntx=1,          ! So our inpcrd file has no velocities

   ! Temperature control
   ntt=3,          ! Langevin dynamics
   gamma_ln=1.0,   ! Friction coefficient (ps^-1) Langevin thermostat
                   ! collision frequency
   tempi=0.0,      ! Initial temp -- give it some small random
                   ! velocities
   temp0=303.15,   ! Target temperature

   ! Potential energy control
   cut=12.0,       ! nonbonded cutoff, in angstroms
   fswitch=10.0,   ! Force-based switching

   ! MD settings
   nstlim=500000, ! 1ns
   dt=0.002,       ! time step (ps)
   ntb=1,          ! Periodic boundaries for constant volume
   ntp=0,          ! No pressure control

   ! SHAKE
   ntc=2,          ! Constrain bonds containing hydrogen
   ntf=2,          ! Do not calculate forces of bonds containing
                   ! hydrogen

   ! Control how often information is printed
   ntpr=100,      ! Print energies every 1000 steps
   ntwx=100,      ! Print coordinates every 5000 steps to the
                  ! trajectory
```

```
   ! Wrap coordinates when printing them to the same unit cell
   iwrap=1,
   ig=-1,         ! Randomize the seed for the pseudo-random number
                  ! generator

   ! Restraint options
   nmropt=1,      ! NMR restraints and weight changes read

/
! Allow the thermostat to change its target temperature throughout
the simulation.
&wt type='TEMP0', istep1=0, istep2=250000, value1=0.0, value2=300.0 /
&wt type='TEMP0', istep1=250001, istep2=500000,
value1=300.0, value2=300.0 /
&wt type='END' /
```

Listing 14: 2_Heat.in

Production

```
&cntrl
   imin=0,         ! No minimization
   irest=1,        ! This IS a restart of an old MD simulation
   ntx=5,          ! So our inpcrd file has velocities

   ! Temperature control
   ntt=3,          ! Langevin dynamics
   gamma_ln=1.0,   ! Friction coefficient (ps^-1) Langevin thermostat
                   ! collision frequency
   temp0=303.15,   ! Target temperature
   ig=-1,          ! Randomize the seed for the pseudo-random number
                   ! generator

   ! Potential energy control
   cut=12.0,       ! nonbonded cutoff, in Angstroms
   fswitch=10.0,   ! Force-based switching

   ! MD settings
   nstlim=50000000, ! 100 ns total
   dt=0.002,        ! time step (ps)

   ! SHAKE
   ntc=2,          ! Constrain bonds containing hydrogen
   ntf=2,          ! Do not calculate forces of bonds containing
                   ! hydrogen

   ! Control how often information is printed
   ntpr=1000,      ! Print energies every 1000 steps
   ntwx=25000,     ! Print coordinates every 25000 steps to the
                   ! trajectory
   ntwr=10000,     ! Print a restart file every 10K steps (can be
```

```
                    ! less frequent)
      ntxo=2,       ! Write NetCDF format
      ioutfm=1,     ! Write NetCDF format (always do this!)

      ! Wrap coordinates when printing them to the same unit cell
      iwrap=1,

      ! Constant pressure control.
      barostat=2,   ! MC barostat... change to 1 for Berendsen
      ntp=1,        ! 1=isotropic, 2=anisotropic, 3=semi-isotropic
      pres0=1.0,    ! Target external pressure, in bar

      ! Set water atom/residue names for SETTLE recognition
      watnam='WAT', ! Water residues are named WAT
      owtnm='O',    ! Water oxygens are named O
   /

   &ewald
      vdwmeth = 0,
   /

   &wt
      type='END'
  /
```

Listing 15: 3_Prod.in

2.3 MoBioTools

At this stage, we have obtained a classical MD trajectory from which we will extract an ensemble of geometries to perform the EDA. To do so, three single point calculations must be carried out for each of the processed geometries as suggested by the supermolecular approach to interaction energies, one corresponding to the interacting complex and the remaining ones, to the isolated fragments. In this tutorial, we will make use of the Gaussian16 program to perform such calculations.

In order to automate the generation of the Gaussian16 input, we will make use of the `MoBioTools` toolkit. To do so, we simply need the `gaussian.main.inp`, on which will specify the amber trajectory and parameters, as well as the number of geometries to be processed, and the Gaussian template, `gaussian.tpl.in`.

```
&main
tpl    = gaussian
top    = tyr_solv.prmtop
traj   = tyr_solv.nc
qmmask = @1-24
geoms = 0 100 2
solvmask = :WAT
closest = 10
```

```
&end
```

Listing 16: gaussian.main.in

```
&header
%NprocShared=1
%mem=2GB
%chk=gau.chk
&end

&route
#P M062X/6-31G(d) charge nosym integral=ultrafinegrid
&end

&externchg !Include MM point charges in the gaussian input
&end

&basis
6-31G(d)
&end

&bsse
mon1 = @1-24
mon2 = @25-12297
&end

&chgspin
0,1,0,1,0,1
&end
```

Listing 17: gaussian.tpl.inp

As one can see from the `gaussian.main.inp` file, we are choosing 50 geometries from the `tyr_solv.nc` trajectory. For the set up of the QM/MM calculation, we have specified a QM region containing tyrosine, as well as the 10 closest water molecules to the amino acid, as specified in the `gaussian.main.inp`. The keyword `bsse` is included in `gaussian.tpl.inp`, to automatically generate the input corresponding to each fragment, as well as that of the complex. The counterpoise approximation is also automatically set to account for the bsse. The `externchg` keyword sets everything out of the QM region as external point charges.

2.4 EDA

Once the generated Gaussian input files have been run, we will be able to perform the EDA. The EDA-NCI program (`https://github.com/marcos-mandado/EDA-NCI`) requires as input the electron density matrices corresponding to the fragments and complex that constitute the interacting systems, which are read from a formatted Gaussian checkpoint file, and a simple template as the one shown here below.

```
mon1    #Name of the Gaussian formatted checkpoitn file for the first
        #monomer
mon2    #Name of the Gaussian formatted checkpoitn file for the second
        #monomer
complex #Name of the Gaussian formatted checkpoitn file for the complex HF
HF      #Level of theory employed for the electron densities. HF also
        #applies to DFT calculations
thre 9  #Threshold employed for the two-electron integrals. N = 9 is the
        #recommended value (10^-N)
```

Listing 18: tmp

However, in order to properly account for the MM part of the system, one must incorporate the MM potential of the fragments and complex into the corresponding nuclear potential of the QM Hamiltonian, the charge section included in the Gaussian input files must be transferred to the corresponding Gaussian formatted checkpoint files using the `charge.exe` utility. To execute it, one must simply type:

```
./charge.exe comp.com mon1.com mon2.com comp.fchk mon1.fchk mon2.fchk
```

Now that the `.fchk` files have been properly modified we can perform the EDA by typing:

```
./EDA.exe $<$ tmp
```

As a result, an output file containing a summary on the main interaction energy components is produced:

```
 ---------------------------------

 Interaction energy terms (in kcal/mol)

  Electrostatic =  -32.553
  Pauli         =   25.912
  Polarization  =  -19.968
  TOTAL         =  -26.588


  Disp+Res-pol  =  -12.514
  Induction     =   -7.455

 ---------------------------------
```

Listing 19: EDA-NCI output summary

Assuming data normality, we can obtain the distribution of each component of the interaction energy based on the sample's mean and standard deviation:

For further details on the usage and installation of EDA-NCI and its associated utilities, please check `https://github.com/marcos-mandado/EDA-NCI`.

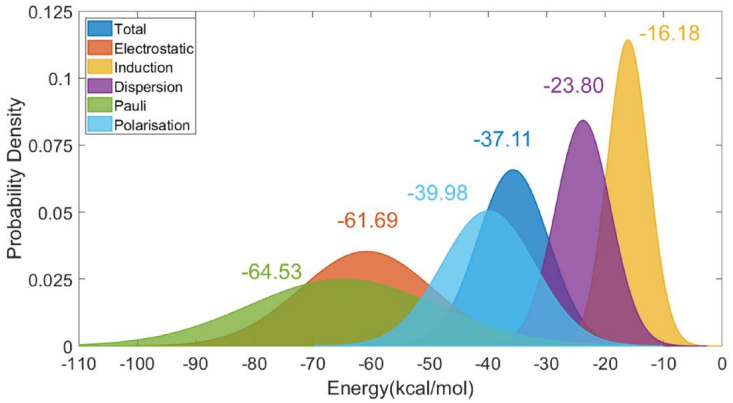


Figure 3: Normal probability distributions for each interaction energy component in kcal/mol obtained from a sample of 50 geometries from a MD trajectory. The Pauli exchange-repulsion is represented with opposite sign. Sample means are reported in kcal/mol

3 Absorption Spectrum: P-diaminoazobenzene Integrated Into The Human Nav 1.4 Channel

In this section, we will use `MoBioTools` to generate input files for calculating the absorption spectrum of a photoswitch bound to a protein ion channel. These calculations will be performed using the QM/MM multistate complete active space second-order perturbation theory (MS-CASPT2) method. To achieve this goal, we will utilize the `pyoverlaps.py` script. This script serves the purpose of generating a Molcas input file for a chosen molecular geometry or a specified range of geometries. Subsequently, it conducts a CASSCF (Complete Active Space Self-Consistent Field) calculation within the Molcas framework. Afterward, it compares the active space orbitals resulting from the CASSCF calculation with those defined in a user-supplied reference active space. The script then performs necessary corrections to ensure that both sets of Molecular Orbitals (MOs) are in alignment, guaranteeing consistency in the computational results.

The system being investigated involves the presence of the p-diaminoazobenzene (p-DAZ) molecule within the voltage-gated ion channel $Na_V1.4$ located in the human brain. The starting setup for this simulation was obtained from a previous research investigation[5]. Following that, an additional 100 nanoseconds of classical molecular dynamics (MD) simulations were carried out to create a collection of molecular configurations. However, for the context of this particular tutorial, we will supply you with both the system's topology and its trajectory data, simplifying your access to the necessary input for further analysis.

3.1 Initial Structures

3.2 The Reference Molecular Orbitals for `pyoverlaps.py`

As the pyoverlaps.py script necessitates a reference set of MOs, our initial focus in this section will be on constructing and executing a geometry optimization procedure. This optimized geometry will serve as the initial structure for conducting a CASSCF single-point energy calcu-

lation. You can construct the molecule using your preferred molecular visualization software, such as Avogadro or IQmol, both of which are open-source and user-friendly. For this tutorial we will only require the free version of Molcas, called OpenMolcas, which can be dowloaded directly from the terminal command:

```
git clone https://gitlab.com/Molcas/OpenMolcas.git
```

For more information, please consult this link:

`https://gitlab.com/Molcas/OpenMolcas`

To begin, we will initiate the optimization of the molecule using the B3LYP/cc-pVDZ level of theory. In the directory tutorialopt you will locate both the non-optimized molecule file named paz.xyz and the Molcas input file designated for geometry optimization, which is named opt.inp:

```
&GATEWAY
Title= P-diaminoazobenzene
coord = paz.xyz; basis = cc-pvdz; group = c1

>>> Do while
  &SEWARD
  &SCF  ;  Title="P-diamino optimization";  KSDFT=B3LYP
  &SLAPAF  &END
>>>  EndDo
```

Listing 20: opt.inp

The location of the `pymolcas.py` script may vary depending on your installation. Regardless of its location, you can execute it using the following command:

```
pymolcas.py opt.inp
```

The geometry optimization process may require some time, but it's essential to obtain a reliable starting point for our reference wave function. Once the optimization is complete, please take the optimized geometry and copy it into the /wrong_12_10 folder. Please exit from the /opt folder and enter in the /wrong_12_10 folder.

In the full active space of p-diaminoazobenzene, you should include all the π and π^* orbitals, as well as the two σ orbitals corresponding to the lone pairs of the central nitrogens. However, for the purposes of this tutorial, we will simplify the active space to include only 4 π orbitals, 2 σ orbitals, and 4 π^* orbitals, totaling 12 electrons in 10 orbitals (12,10). Regrettably, performing a single CASSCF single point calculation doesn't always yield the expected active space, which is also the reason for the development of the pyoverlaps.py script. Inside the folder you'll see an input file called wrong_12_10.inp:

```
&GATEWAY
Title= Acrolein molecule
coord = opt.xyz; basis = CC-PVDZ; group = c1
&SEWARD;
&SCF
&RASSCF
```

```
LumOrb
nActEl = 12 0 0;
Inactive = 50; Ras2 = 10 ;
Symmetry = 1;
Spin = 1;
CiRoot = 10 10 1
```

Listing 21: wrong_12_10.inp

We will now conduct a series of calculations. First, we will initiate a Hartree-Fock (HF) calculation, followed by a subsequent CASSCF single-point calculation involving 12 electrons distributed among 10 orbitals. The CASSCF calculation will read the HF orbitals for its computation. Please execute this sequence of calculations as follows:

```
pymolcas.py wrong_12_10.inp
```

Please open the *.molden file and look at the active orbitals. In the unique scenario involving p-diaminoazobenzene, there's a noteworthy aspect to consider. If you generate your MOs using the HF method and subsequently employ them as a checkpoint for a CASSCF single-point calculation, the resulting MOs may not align with your expectations, since one of the two σ orbitals in not present. To address this issue, two potential strategies emerge.
Firstly, you can explore the option of directly rotating the orbitals derived from the HF calculation. This approach involves manipulating the existing HF MOs to align them more closely with the desired electronic structure. Alternatively, if orbital rotation doesn't yield the desired outcome, you may consider expanding the active space encompassing the occupied orbitals. By increasing the active space of the occupied orbitals, you provide the calculation with a broader selection of orbitals from which it can construct the desired MOs. This expanded active space may facilitate the inclusion of the target MOs, simplifying subsequent orbital manipulations.
Now, we will proceed to run a Hartree-Fock (HF) calculation, followed by a CASSCF calculation with an active space of (16,10) for a single-point energy calculation. To do this, please exit the current folder and enter the 16_10 folder. Within this folder, you will find the optimized geometry and an input file named 16_10.inp:

```
&GATEWAY
Title = P-diaminoazobenzene
coord = opt.xyz; basis = CC-PVDZ; group = c1
&SEWARD;
&SCF
&RASSCF
LumOrb
nActEl = 16 0 0;
Inactive = 48; Ras2 = 10 ;
Symmetry = 1;
Spin = 1;
CiRoot = 10 10 1
```

Listing 22: 16_10.inp

Please execute this sequence of calculations as follows:

```
pymolcas.py 16_10.inp
```

Upon reopening the *rasscf.molden file, you may notice the presence of one of the σ orbitals, which was previously absent in the HF single-point calculation. This orbital now occupies position 49. However, in the case of a (12,10) calculation, it lies outside the active space. Thus, if we attempt to employ the CASSCF (16,10) orbitals directly for the (12,10) single-point calculation, this particular orbital would be excluded. Therefore, it becomes necessary to perform an orbital rotation, and this can be achieved using one of the higher-energy orbitals. To do this, please exit the current folder and enter the 12_10 folder, where you will find the following input, 12_10.inp

```
&GATEWAY
Title = P-diaminoazobenzene
coord = opt.xyz; basis = CC-PVDZ; group = c1
&SEWARD;

>>>COPY $CurrDir/16_10.RasOrb $WorkDir/INPORB

&RASSCF
LumOrb
nActEl = 12 0 0;
Inactive = 50; Ras2 = 10 ;
Symmetry = 1;
Spin = 1;
CiRoot = 10 10 1
alter = 1
1 49 52
```

Listing 23: 12_10.inp

In order for the calculation to proceed, please copy the `*.RasOrb` file from the `16_10` folder into the current `12_10` folder, which will allow the program to read the orbitals from the (16,10) calculation. Please execute this sequence of calculations as follows:

```
pymolcas.py 12_10.inp
```

Open the `*rasscf.molden` file and look at the orbitals again. If everything proceeded well, you should have 4 π, 2 σ and 4 π^* orbitals which we will be using for the QM/MM calculation. Please copy the `*rasscf.molden` file into the `tutorial3/pyoverlaps/casscf` folder.

3.3 The `pyoverlaps.py` Program

After this extensive process, you can now utilize the `12_10.molden` file as a reference file for the `pyoverlaps.py` script. As previously mentioned, the desired active space should consist of 4 π orbitals, 2 σ orbitals, and 4 π^* (pi star) orbitals, resulting in a total of 12 electrons distributed among 10 orbitals, denoted as (12,10). Inside the folder, you'll discover the following files: system.parm7, which contains topology data; 100f.nc, a trajectory file comprising 100 frames; system.pdb, enabling you to review ligand and residue numbering; and molcas.inp which is a MOLCAS input file:

```
&GATEWAY
Title = p-diaminoazobenzene
Coord = azo.xyz
Group = C1
Basis set = cc-pvdz spherical all
XFIeld = charges.dat
&SEWARD
&SCF
&RASSCF
Title = rasscf
Symmetry = 1
Spin = 1
Nactel = 12 0 0
Inactive = 50
Ras2 = 10
CiRoot = 10 10 1
```

Listing 24: molcas.inp

The input for the current CASSCF calculation, which corresponds to the (12,10) active space, should remain identical to the previous calculation. In addition, it includes the "XField" keyword to read the point charge data, which you can extract using the pyoverlaps.py script. Now, you can proceed to execute the pyoverlaps.py script with the following command:

```
pyoverlaps.py -p system.parm7 -c 100f.nc -a true -r 12_10.molden
-tpl molcas.inp -rng 51 60 -cl yes -qm :LIG -n 100
```

The program will repeatedly execute CASSCF calculations until the active space of the geometry in question matches the reference. This "correction" of the active space is accomplished through the utilization of the algorithm outlined in reference [6], in conjunction with the `alter` keyword within the &RASSCF module of Molcas. Once all the geometries have been processed, a file named "results.dat" will emerge in the directory. This file will display the exchanged MOs (MOs) for every individual geometry. If the iteration count reaches 5, it implies that the program could not successfully rotate the MOs. However, in our scenario, 72 out of 100 geometries have achieved convergence. With the attainment of the accurate wave function at CASSCF (12,10), we can now proceed to execute MS-CASPT2 calculations and acquire reasonably accurate energy values. Let's leave the `/casscf` directory and navigate to the `/caspt2` folder. Within this directory, we will utilize the `main_qminputs.py` script to generate the CASPT2 inputs necessary for the upcoming QMMM (Quantum Mechanics/Molecular Mechanics) calculation. Notice that this is the script used in the other tutorials to generate the corresponding input files. `main_qminputs.py` relies on two existing files in the folder, one of which is `geom.inp`:

```
&main
tpl    = molcas
top    = system.parm7
traj   = 100f.nc
qmmask = :587
geoms = 0 100 1
```

```
&end
```

Listing 25: geom.inp

It reads in the software for interfacing, topology data, trajectory information, and provides options to specify the QM region and the number of geometries to generate. Additionally, you will find the MOLCAS-specific file named `molcas.tpl` in the same folder:

```
&header
title = p-AZO molecule
group = C1
basis set = cc-pVDZ spherical all
RICD
&end

&seward
&end

&chgspin
0,1,0,1,0,1
&end

&externchg
&end

&guess
jobiph
&end

&caspt2
IPEA= 0.0
SHIFT= 0.0
IMAGinary= 0.2
maxiter=100
multistate = 10 1 2 3 4 5 6 7 8 9 10
PROP
&end

&rassi
Nr of JobIph = 1 10; 1 2 3 4 5 6 7 8 9 10
HEFF
&end
```

Listing 26: molcas.tpl

This script is designed to supply all the essential keywords required to execute a successful CASPT2 single-point calculation. Please execute it as:

```
main_qminputs.py -i geom.inp -t molcas.tpl
```

which will generate 100 geometries with the `.xyz` file of the QM region, the MM region and the MOLCAS input file. However, there is a critical component that is currently missing: the

CASSCF reference wave function. To rectify this, we can implement a loop that copies the `*.JobIph` file from each geometry in the `/casscf` folder to the corresponding geometry in the `/caspt2` folder. This step is essential to ensure that the CASPT2 calculations have access to the necessary CASSCF wave function information.

```
for i in {1..100}
do
cp casscf/geom$i/*.JobIph caspt2/geom$i/geom$i.in.guess.JobIph
done
```

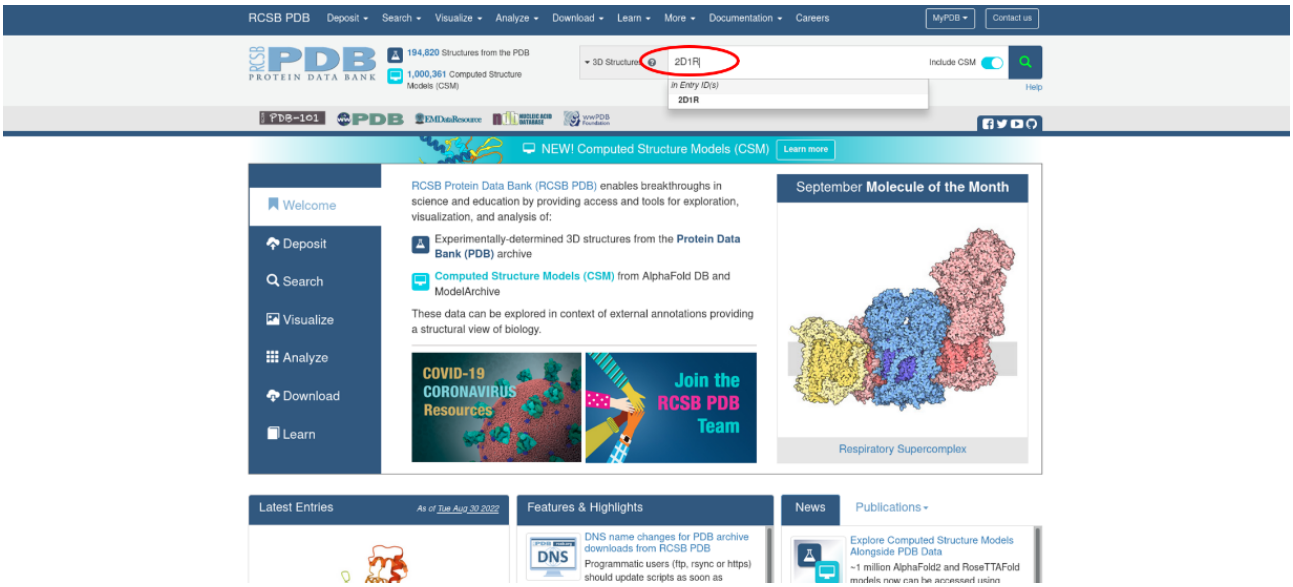
Please run the calculation with `pymolcas.py` in the corresponding folder of each geometry. If everything proceeded well, you should now have a MS-CASPT2 single point for each geometry in which the orbital rotation of the `pyoverlaps.py` was successful. This is the end of the tutorial.

4 ABSORPTION SPECTRUM: LUCIFERINE/LUCIFERASE COMPLEX

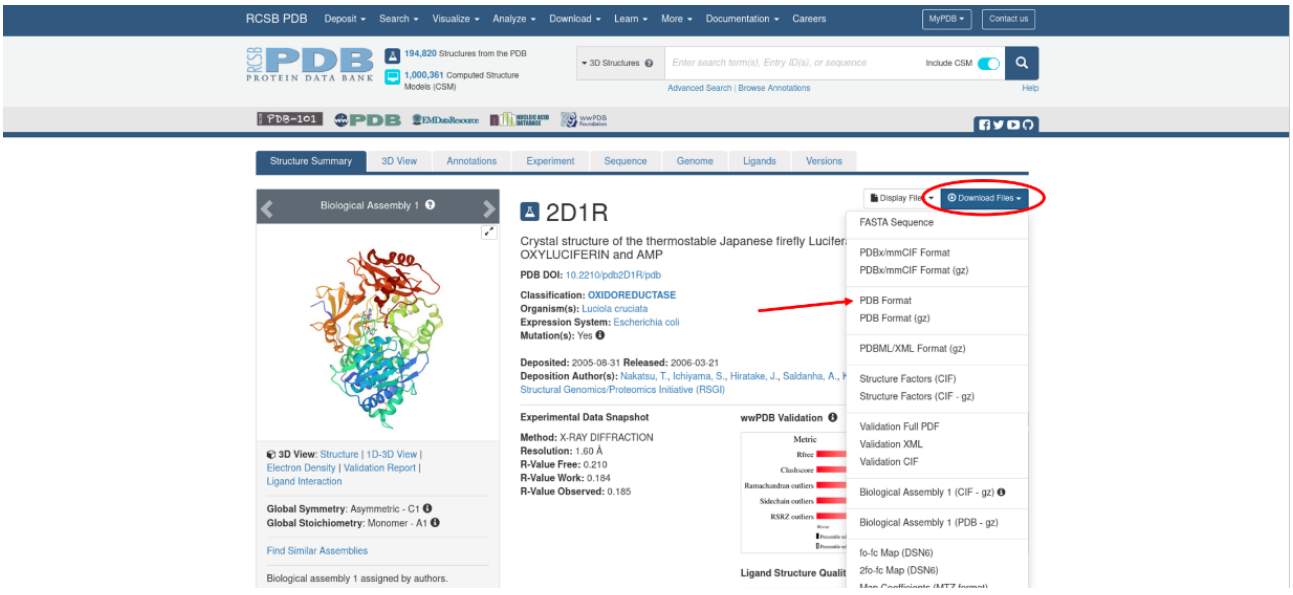
4.1 Setup

4.1.1 Structure of the Protein

In this tutorial, we will simulate the absorption spectrum of the Luciferine/Luciferase complex, whose crystallographic structure (PDB code: 2D1R) can be retrieved from the e Protein Data Bank website (`https://www.rcsb.org/` , Figure 4). We will use `PyMOL` to pre-process it prior to generate the actual input files with `MoBioTools`. The PDB structure can either be retrieved from the previously mentioned website, or directly downloaded using the `PyMOL` graphical user interface by means of the `fetch` command (Figure 5).



(a) Homepage of the PDB.



(b) PDB ID: 2D1R section of the PDB.

Figure 4: Screenshots of the PDB.

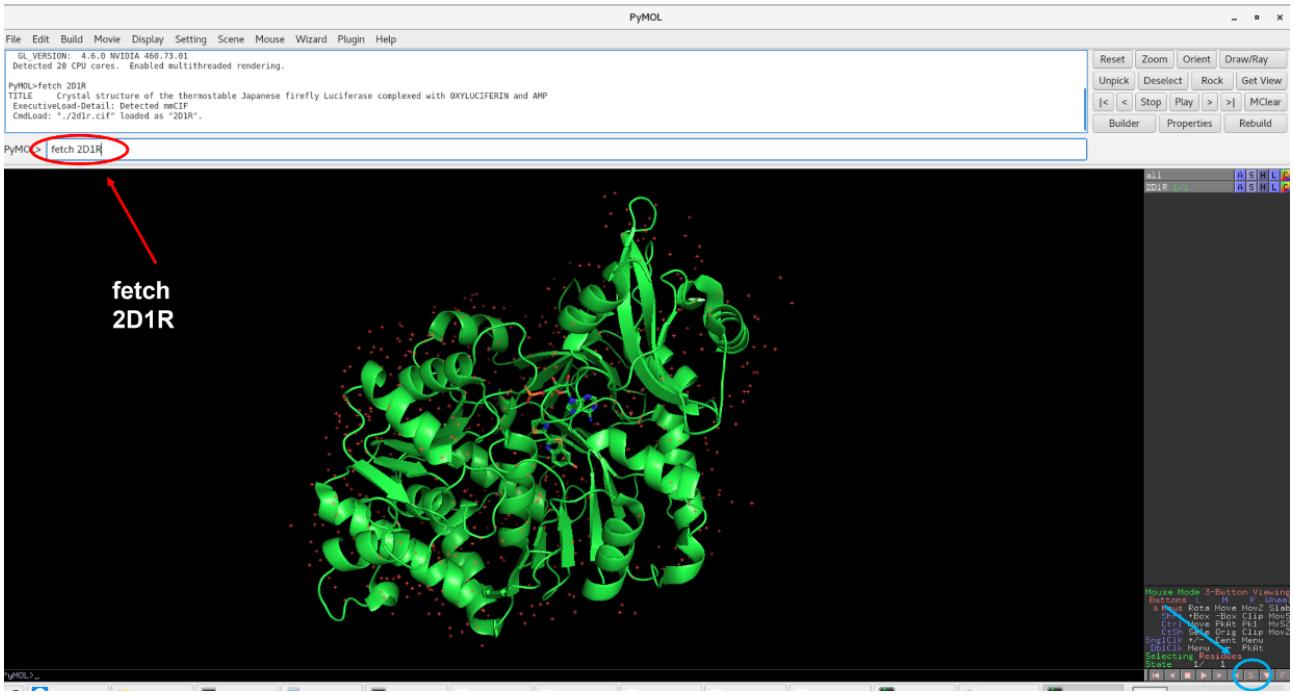


Figure 5: Preview of the 2D1R PDB structure with PyMOL.

As can be observed from Figure 5, the obtained structure contains some water molecules that need to be removed since a latter step involves the solvation of the whole system. For this, we need to display the sequence of amino acids and residues by clicking on the *S* letter circled in blue at the bottom-right of Figure 5. Then, we look for the sequence of oxygen atoms, select it and remove the atoms of the selection. Note that there are no hydrogen atoms in the structure, since the experimental technique does not allow their detection. To select the atoms, click on the first `O`, as shown in Figure 6; slide the underlying bar until the last atom appears, press `SHIFT` on the keyboard and select everything until the end by clicking on the last atom. Then, these atoms can be easily removed by clicking in the `A` (action) of the `sele` section on the left and clicking on `remove atoms`, as shown in Figure 6.

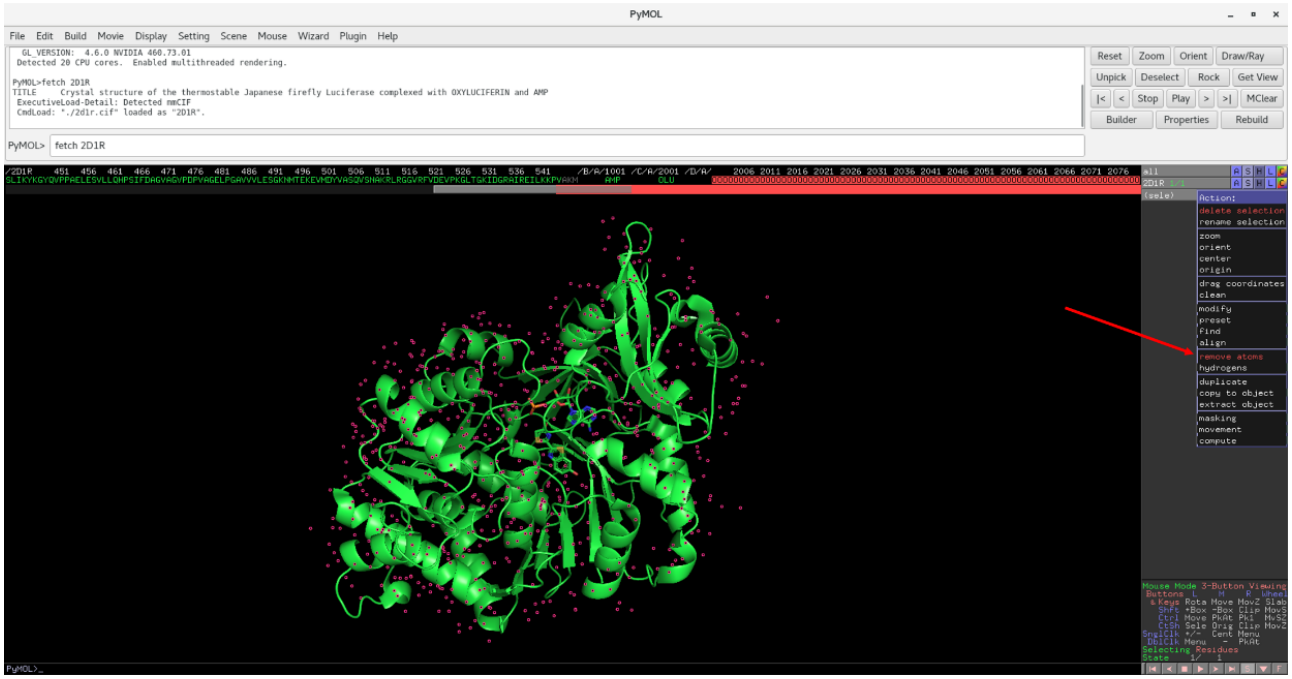


Figure 6: Selecting and removing residues of a PDB structure with PyMOL.

As a result, the structure on Figure 7 is obtained. Nevertheless, this structure is still missing the hydrogen atoms. The H atoms corresponding to the protein will be automatically added later on in the process, whereas those of the oxyluciferine (OLU) and the adenosine monophosphate (AMP) need to be added manually. To do so, please select the OLU and AMP residues in the upper sequence section, click on the `A` of the `sele` section on the left and then click on `hydrogen atoms` > `add`, as shown in Figure 7. This would generate the monoanion of the AMP and the phenol-enolate tautomer of the oxyluciferine chromophore.

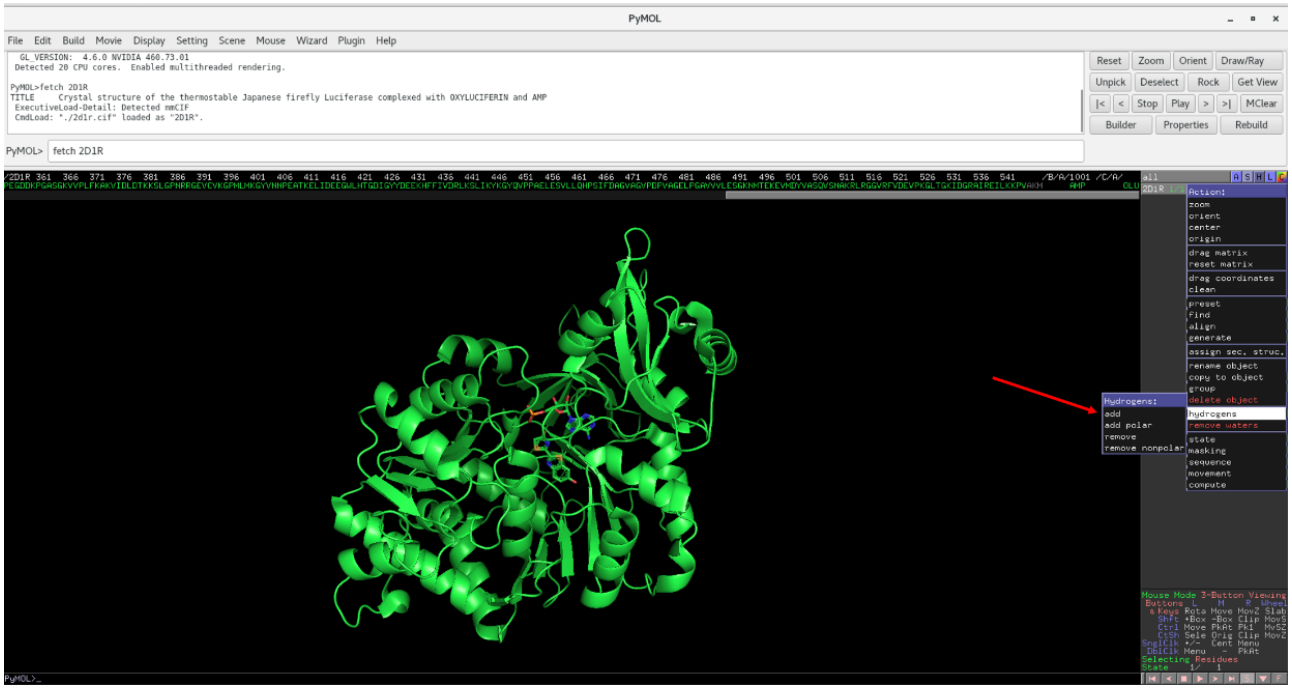


Figure 7: Protonation of a structure with PyMOL.

To properly save it, the whole structure needs to be copied to a new object. To do so, select the residues of the protein by choosing the `chain` selection at the bottom-right of the window, as shown in Figure 8.

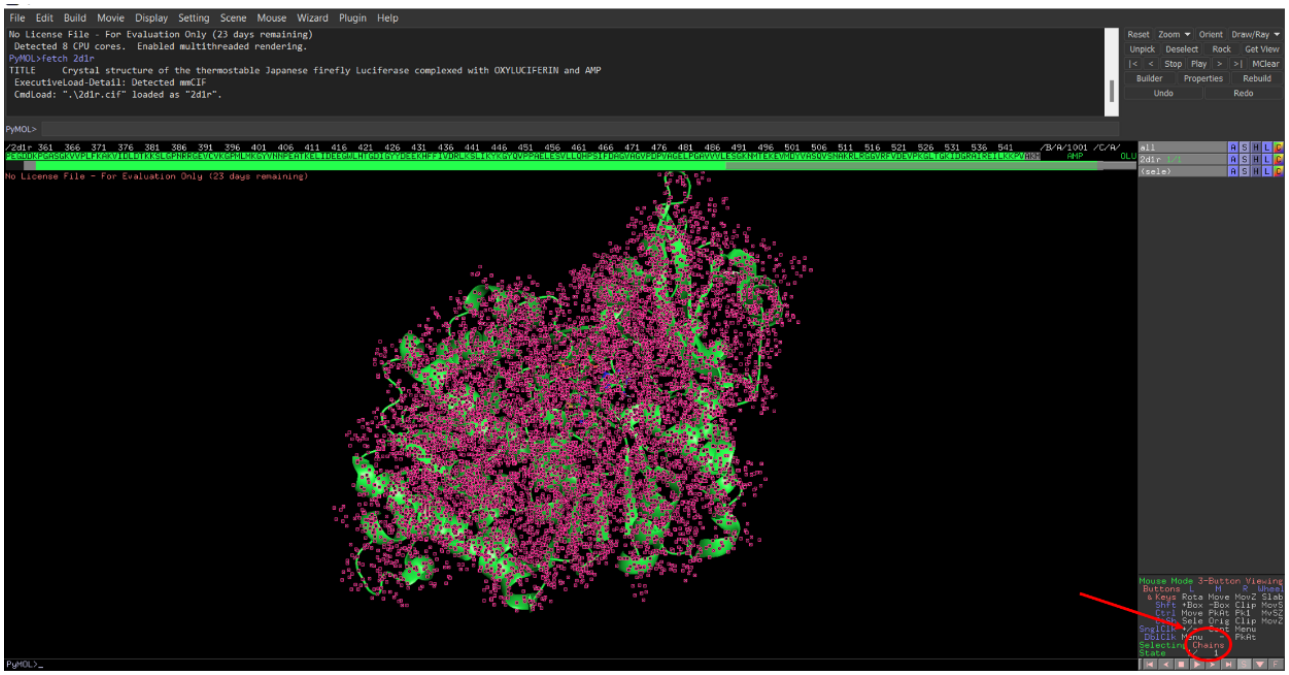


Figure 8: Enabling chain the selection option in PyMol.

Subsequently, click on the `A` letter of the selection > copy to object > new (see Figure 9).

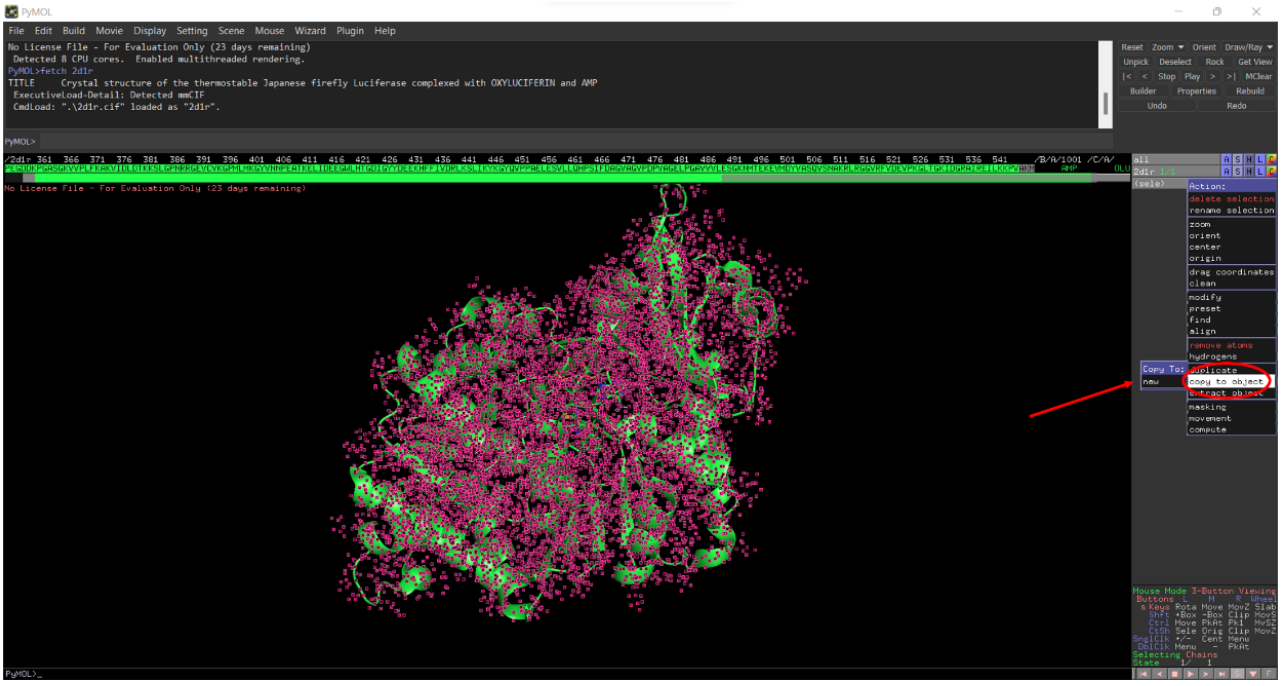


Figure 9: Adding a selection to a new object in PyMol.

By selecting the AMP residue and following the same procedure, we can add the selection to the previously generated object if, instead of selecting a new object, we select `obj01`. Re-

peating the same procedure for the OLU, then the new object will encompass the whole system.

Finally, save the PDB file (`system2.pdb`) by click on `File` > `Export Molecule` on the top-left and then, mark the option `original atom order`, as shown in Figure 10.

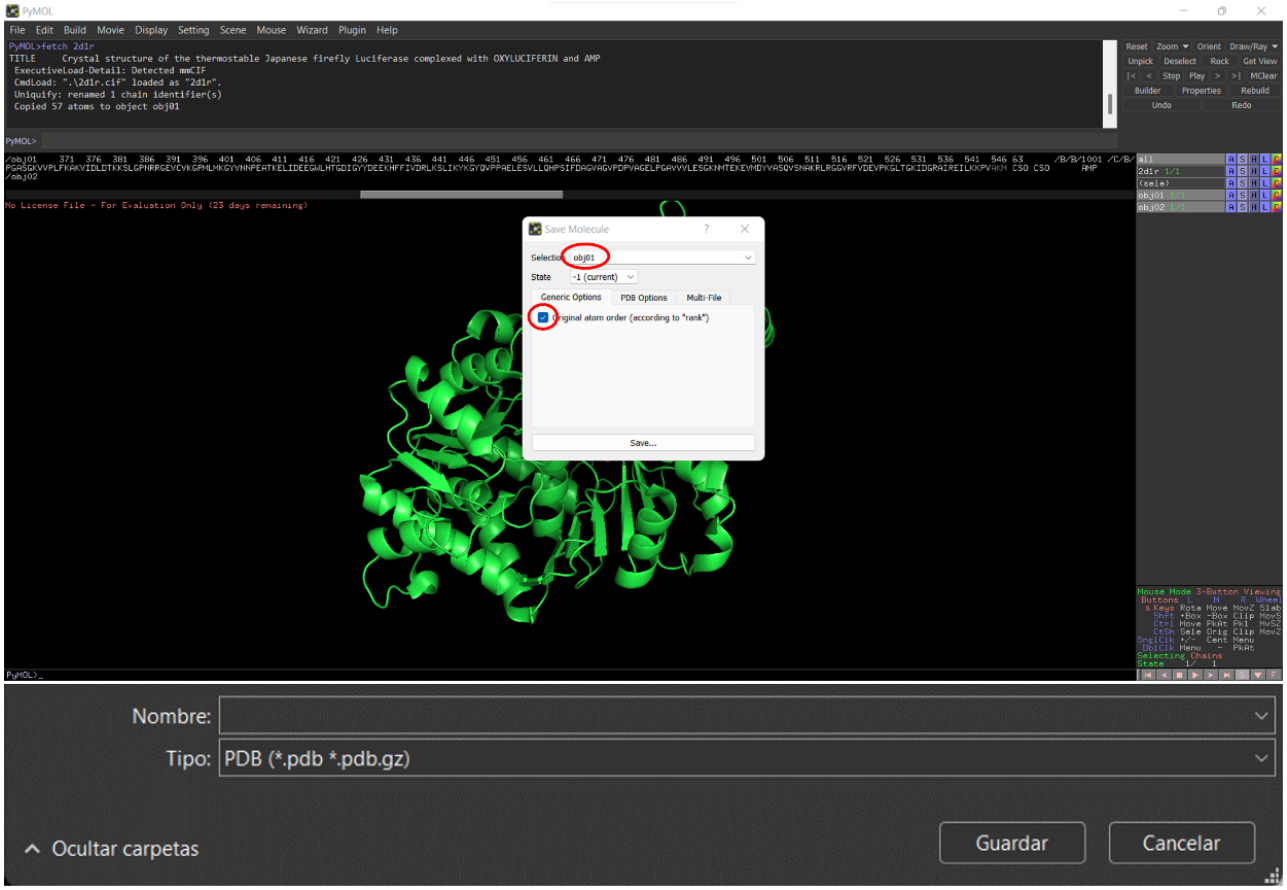


Figure 10: Saving options.

Additionally, we will copy the AMP and OLU residues to objects by individually selecting them and clicking again in `A` of the `sele` section > `copy to object` > `new`, and save both `obj02` and `obj03` as two PDB files since we will need them later on.

Some extra changes need to be done in the new PDB file so as to have it in a format compatible with `antechamber`, which will be used in the following section. Open the file with any text editor, for example `vi`:

```
vi system.pdb
```

Inside the file, look for the residues `CSO 63` and `CSO 64`. For each of these residues, perform the following changes:

- `CSO` by `CYS`
- `OD` by `HG`

- `HETATM` by `ATOM`.

Moreover, for both the OLU and AMP residues we have to change:

- `HETATM` by `ATOM`.
- Atom type + atom number in the atom name column, the third one as shown in Figure 11.

```
CRYST1   57.612  181.402   52.743  90.00  90.00  90.00 P 21 21 21    4
ATOM      1  O1  OLU A2001      16.670  26.100  10.887  1.00 32.06       C    O
ATOM      2  C2  OLU A2001      15.028  24.679   9.909  1.00 30.05       C    C
ATOM      3  C3  OLU A2001      16.429  25.049  10.303  1.00 31.21       C    C
ATOM      4  N4  OLU A2001      17.314  24.123   9.956  1.00 30.21       C    N
ATOM      5  C5  OLU A2001      16.764  23.072   9.331  1.00 30.07       C    C
ATOM      6  S6  OLU A2001      15.145  23.066   9.100  1.00 32.70       C    S
ATOM      7  S7  OLU A2001      19.260  22.005   9.098  1.00 26.21       C    S
ATOM      8  C8  OLU A2001      17.572  21.905   8.870  1.00 28.51       C    C
ATOM      9  N9  OLU A2001      17.122  20.761   8.333  1.00 26.03       C    N
ATOM     10  C10 OLU A2001      18.079  19.867   8.068  1.00 26.39       C    C
ATOM     11  C11 OLU A2001      19.363  20.481   8.469  1.00 26.70       C    C
ATOM     12  C12 OLU A2001      20.526  19.726   8.302  1.00 27.50       C    C
ATOM     13  C13 OLU A2001      20.463  18.435   7.744  1.00 27.89       C    C
ATOM     14  O14 OLU A2001      21.598  17.690   7.564  1.00 29.92       C    O
ATOM     15  C15 OLU A2001      19.241  17.879   7.363  1.00 26.46       C    C
ATOM     16  C16 OLU A2001      18.054  18.591   7.519  1.00 23.27       C    C
ATOM     17  H17 OLU A2001      14.124  25.264  10.077  1.00 30.05       C    H
ATOM     18  H18 OLU A2001      21.488  20.139   8.605  1.00 27.50       C    H
ATOM     19  H19 OLU A2001      21.852  17.711   6.638  1.00 29.92       C    H
ATOM     20  H20 OLU A2001      19.215  16.876   6.938  1.00 26.46       C    H
ATOM     21  H21 OLU A2001      17.107  18.148   7.211  1.00 23.27       C    H
CONECT    1    3
CONECT    2   17    3    6
```

Figure 11: Changed atom name column.

These changes need to be done not only in the PDB file containing the whole system but also in the the OLU and AMP separated PDB files.

4.1.2 Solvation and Parameters

The Amber software already has predefined force field parameters (equilibrium values and force constants) to deal with proteins and water. However, the luciferin/luciferase complex contains two non-standard residues: AMP and OLU, for which it is necessary to determine extra parameters. The procedure is analogous for both of them, so we will describe in detail the case of OLU. At first, we transform the `OLU.pdb` file into a `.mol2` file :

```
\textit{antechamber -i OLU.pdb -fi pdb -o OLU.mol2 -fo mol2 -c bcc -at gaff
    -nc -1}\\
```

where the options `-i`, `-fi`, `-o`, `-fo`, `-c`, and `-nc` correspond to the input name, input format, output name, output format, method to determine the charges, and the total charge of the system, respectively.

Then, we have to check whether there are parameters for the molecule in the selected force field and apply the proper corrections in the case that there aren't any:

```
parmchk2 -i OLU.mol2 -o OLU.frcmod -f mol2
```

where the options `-i`, `-o`, and `-f` indicate the input file, the output file and the input format, respectively. As stated above, this process needs to be repeated in the case of the AMP molecule:

```
antechamber -i AMP.pdb -fi pdb -o AMP.mol2 -fo mol2 -c bcc -at gaff -nc -1

parmchk2 -i AMP.mol2 -o AMP.frcmod -f mol2
```

Afterwards, we have to generate the library files for the ligands by means of tleap:

```
tleap -f lib.in
```

where the input file `lib.in` content is explained below:

```
source leaprc.gaff2        # source leaprc file for gaff

OLU=loadmol2 OLU.mol2      # load mol2 file of the oxyluciferine
saveoff OLU OLU.lib        # create library of the oxyluciferine

AMP=loadmol2 AMP.mol2      # load mol2 file of the AMP
saveoff AMP AMP.lib        # create library of the AMP

quit
```

Listing 27: lib.in

Finally, we will generate the coordinates and the parameter files to be used during the MD simulations using again the `tleap` program with the `solvation.in` input file:

```
tleap -f solvation.in

source leaprc.gaff2                # source leaprc file for gaff
source leaprc.protein.ff19SB       # source leaprc file for ff19SB
                                     protein force field
source leaprc.water.tip3p          # source leaprc file for TIP3P
                                     water model
loadamberparams OLU.frcmod         # load OLU parameters
loadamberparams AMP.frcmod         # load AMP parameters

loadoff OLU.lib                    # load library for OLU
loadoff AMP.lib                    # load library for AMP
SYS=loadpdb system2.pdb            # load whole system

solvateOct SYS TIP3PBOX 5.0        # solvate system with H2O with
                                     truncated octaedron box of 5 A
addIons2 SYS Na+ 0                 # add ions - cell net charge
                                     must be 0
```

```
savepdb SYS final_system.pdb       # save pdb of the whole system
saveamberparm SYS final_system.prmtop final_system.rst7
                                    #save pamerters

quit
```

Listing 28: solvation.in

If one opens the final PDB file, it is possible to observe the solvated system and the truncated octahedron shape of the cell.

4.2 Minimization, Equilibration and Production

Once the `.prmtop` (force field parameters) and the `.rst7` (coordinate) files are prepared, the system is ready for us to perform the MD simulations. The inputs for the minimization, heating and equilibration/production simulations are shown below.

```
    Minimization input file in explicit solvent
 &cntrl
    ! Minimization options
    imin=1,         ! Turn on minimization
    ntx=1,          ! Read position but not velocities
                    !(5 for reading also velocities)
    irest=0,        ! Do not restart (=1) the simulation
    maxcyc=10000,   ! Maximum number of minimization cycles
    ncyc=5000,      ! Method switched from steepest descent to
                     !conjugate gradient after NCYC
    ntb=1,          ! PBC cte V
    iwrap=1,

    ! Potential energy function options
    cut=9.0,        ! Nonbonded cutoff in Angstroms
    fswitch=7.0,    ! Van der Waal cutoff

    !Restraint options
    nmropt=1,       !Restraints

    ! Control how often information is printed to the output file
    ntpr=100,       ! Print energies every 100 steps
    ntwx=0,         ! No trajectory file
 /
&wt type = 'END' /
```

Listing 29: 01_Min_luc.in

```
    Heat
 &cntrl
  imin=0,         ! Turn off minimization
  ntx=1,          ! Read position but not velocities (5 for reading
                  !also velocities)
```

```
  irest=0,         ! Do not restart (=1) the simulation

  nstlim=50000,    ! Perform MD for 50000 steps
  dt=0.002,        ! 2 fs time step
  ntf=2, ntc=2,    ! Shake hydrogen atoms
  tempi=0.0,       ! Initial temperature
  temp0=300.0,     ! Target temperature

  ! Control how often information is printed to the output file
  ntpr=100,        ! Energy information will be printed every ntpr steps
  ntwx=100,        ! Coordinates written to the mdcrd file
                   !every ntwx steps

  ! Potential energy function options
  cut=9.0,         ! Nonbonded cutoff in Angstroms
  fswitch=7.0,     ! Van der Waal cutoff

  ! Ensemble control
  ntb=1,           ! PBC cte V
  ntp=0,           ! No pressure scaling (Default)
  ntt=3,           ! Langevin dynamics
  gamma_ln=2.0,    ! Collision frequency for the Langevin
                   !dynamics in ps-1

  ! Wrap coordinates when printing them to the same unit cell
  nmropt=1,
  iwrap=1,

  ig=-1,           ! Random seed to use to generate velocity distribution
 /
! Change conditions during simulation
! Simulation increases T from 0 to 300K during steps 0 to 25000
&wt type='TEMP0', istep1=0, istep2=25000, value1=0.0, value2=300.0 /
! Keep T cte for the rest of the simulation
&wt type='TEMP0', istep1=250001,
istep2=50000, value1=300.0,
value2=300.0 /
&wt type='END' /
```

Listing 30: 02_Heat_luc.in

```
    A NPT simulation for common production-level simulations
 &cntrl
    imin=0,           ! No minimization
    irest=1,          ! This IS a restart of an old MD simulation
    ntx=5,            ! So our inpcrd file has velocities

    ! Temperature control
    ntt=3,            ! Langevin dynamics
    gamma_ln=1.0,     ! Friction coefficient (ps^-1)
```

```
   temp0=303.15,     ! Target temperature
   ig=-1,            ! Seed for velocities

   ! Potential energy control
   cut=9.0,          ! nonbonded cutoff, in Angstroms
   fswitch=7.0,      ! Force-based switching

   ! MD settings
   nstlim=10000000, ! 20ns total
   dt=0.002,         ! time step (ps)

   ! SHAKE
   ntc=2,            ! Constrain bonds containing hydrogen
   ntf=2,            ! Do not calculate forces of bonds containing
                     !hydrogen

   ! Control how often information is printed
   ntpr=1000,        ! Print energies every 1000 steps
   ntwx=25000,       ! Print coordinates every 25000 steps to the
                     !trajectory
   ntwr=10000,       ! Print a restart file every 10K steps (can be
                     !less frequent)
   ntxo=2,           ! Write NetCDF format
   ioutfm=1,         ! Write NetCDF format (always do this!)

   ! Wrap coordinates when printing them to the same unit cell
   iwrap=1,
   nmropt=1,

   ! Constant pressure control.
   barostat=1,       ! Berendsen barostat
   ntp=1,            ! 1=isotropic, 2=anisotropic, 3=semi-isotropic
                     !w/ surften
   pres0=1.0,        ! Target external
                     !pressure, in bar

   ! Set water atom/residue names for
   !SETTLE recognition
   watnam='WAT',     ! Water residues
                     !are named WAT
   owtnm='O',        ! Water oxygens
                     !are named O
/

&ewald
   vdwmeth = 0,
/

&wt
```

```
    type='END'
/
```

Listing 31: 03_Prod_luc.in

4.3 MoBioTools

Once the molecular dynamics simulation is finished, we will have to extract some geometries out of the trajectory to compute the absorption spectrum of the luciferine/luciferase complex. To do so, we will use the `MoBioTools` software that automatically takes the geometries and prepares the inputs. For this, we will need two files: the `gaussian_luc.main.in`, that specifies the parameter and trajectory files to be read and the quantum-mechanical region, among other things; and the Gaussian template file, the `gaussian_luc.tpl.inp`. In general, the script that prepares the quantum-mechanical input files can be run in the command line as:

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

```
&main
tpl    = gaussian
top    = final_system.prmtop
traj   = 03_prod.nc
qmmask = :OLU
geoms = variabless
solvmask =:WAT
&end
```

Listing 32: gaussian_luc.main.in

```
&header
%NprocShared=4
%mem=16GB
&end

&route
#p B3LYP/6-31G NoSym TD(singlets,nstates=10)
&end

&externchg
&end

&chgspin
-1,1
&end
```

Listing 33: gaussian_luc.tpl.inp

However, for the generation of many geometries one can use the script shown below:

```
#!/bin/bash
```

```
for number in ` seq 199 2 399` ; do
   sed -i "s/variabless/$number/g" gaussian.main.in
   main_qminputs.py -i gaussian.main.in -t gaussian.tpl.inp
   sed -i "s/$number/variabless/g" gaussian.main.in
done
```

Listing 34: many_geoms_luc.sh

4.4 Absorption Spectrum

Once the quantum-mechanical calculations are finished, we will retrieve the energies involved in the calculated electronic transitions. A simple script for this can be found below

```
#!/bin/bash
apwd=`pwd`

echo '# geom   state   Eexc (eV)  (nm)  f' > ${apwd}/SUMMARY_STEPS.dat

for i in `seq 199 2 399`
do
 for j in `seq 9`
   do
   exc=`grep "Excited State   $j" geom${i}/geom${i}.log | awk '{print $5}'`
   f=`grep "Excited State   $j" geom${i}/geom${i}.log | awk '{print $9}'`
   nm=`grep "Excited State   $j" geom${i}/geom${i}.log | awk '{print $7}'`
   echo "   ${i}   $j    $exc   ${f#*=}   $nm" >> ${apwd}/SUMMARY_STEPS.dat
   done
   exc=`grep "Excited State  10" geom${i}/geom${i}.log | awk '{print $5}'`
   f=`grep "Excited State  10" geom${i}/geom${i}.log | awk '{print $9}'`
   nm=`grep "Excited State  10" geom${i}/geom${i}.log | awk '{print $7}'`
   echo "   ${i}   10    $exc   ${f#*=}   $nm" >> ${apwd}/SUMMARY_STEPS.dat
done
```

Listing 35: get_energies.sh

With these energies and their corresponding intensities, one can place a Gaussian function on top of each of them in order to obtain the absorption spectrum using, for example the python script shown below:

```
import numpy as np
                                                        import pandas as
   pd
from matplotlib import pyplot as plt
import math

def gaussian (x, FWHM, i , f):
    sigma = FWHM/2.355
    return np.exp(-np.power((x - i), 2.)/(2*sigma))*f

#FITTING A GAUSSIAN TO EACH LINE
```

```
x = np.linspace(0.5, 6.0, 300)

archivo="SUMMARY_STEPS.dat"
out="spectrum_"+archivo
out2="spectrum_nm_"+archivo
data= np.loadtxt(archivo)
ro=data[:,2]
fo=data[:,3]

gaus = np.zeros(len(x))
for i in range(0,len(ro)):
    gaus += gaussian(x, 0.3, ro[i], fo[i])
with open(out, 'w') as f:
    f.write('#E (eV)    Intensity')
    f.write('\n')
    for i in range(len(x)):
        f.write(str(x[i]))
        f.write('   ')
        f.write(str(gaus[i]))
        f.write('\n')
with open(out2, 'w') as f:
    f.write('#E (nm)    Intensity')
    f.write('\n')
    for i in range(len(x)):
        f.write(str((10000000*27.211/219474)/x[i]))
        f.write('   ')
        f.write(str(gaus[i]))
        f.write('\n')
```

Listing 36: spectrum.py

Finally, the generated file can be represented con any plotting utility. The script `plot_absorption.gpl` shows a `gnuplot` example on how to do so.

```
set term png
set output 'plot.spectrum_SUMMARY_STEPS.png'

set key font ",20" spacing "1.5"

set xtics font ",15"
set ytics font ",15"

unset key

set xlabel "Absorption Energy (eV)" font ",20"
set ylabel "Intensity" font ",20" offset 2,0,0

p 'spectrum_SUMMARY_STEPS.dat' u 1:2 w l
lc rgb "#D52D00" lw 3
```

Listing 37: plot.absorption.gpl

The script can be executed as follows:

```
gnuplot plot.absorption.gpl
```

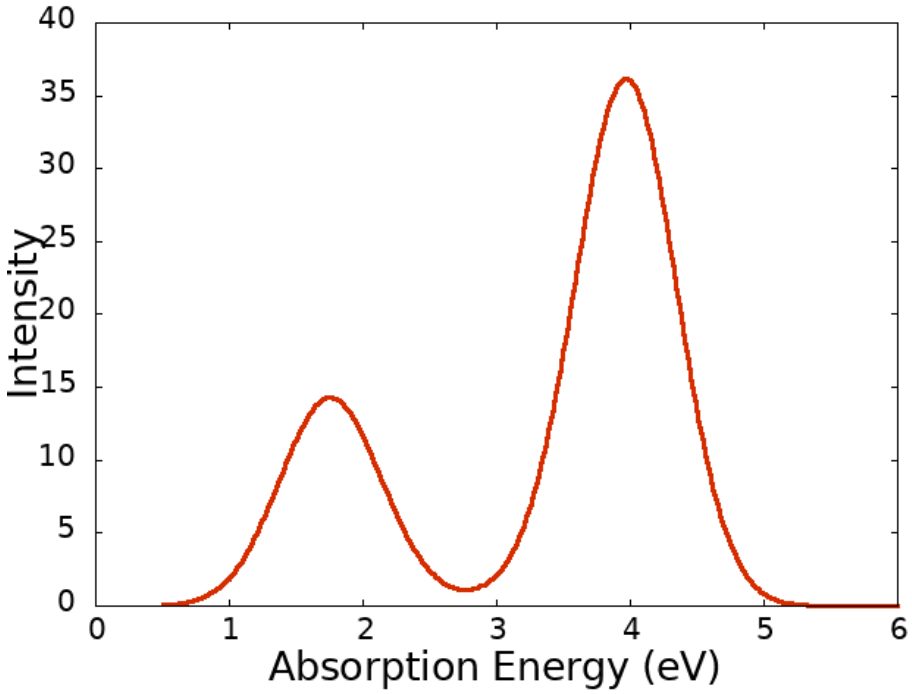


Figure 12: Absorption spectrum of the luciferine/luciferase complex

And with this, we come to the end of this tutorial.

References

[1] N. Ramos-Berdullas, I. Pérez-Juste, C. Van Alsenoy and M. Mandado. Theoretical study of the adsorption of aromatic units on carbon allotropes including explicit (empirical) DFT dispersion corrections and implicitly dispersion-corrected functionals: the pyridine case. *Phys. Chem. Chem. Phys.*, 17(1):575–587, 2015.

[2] M. Mandado and J. M. Hermida-Ramón. Electron Density Based Partitioning Scheme of Interaction Energies. *J. Chem. Theory Comput.*, 7(3):633–641, 2011.

[3] G. Cárdenas, A. Pérez-Barcia, M. Mandado, and J. J. Nogueira. Characterization of cis-platin/membrane interactions by QM/MM energy decomposition analysis. *Phys. Chem. Chem. Phys.*, 23:20533, 2021.

[4] M. Ropo, M. Schneider, C. Baldauf, and V. Blum. First-principles data set of 45,892 isolated and cation-coordinated conformers of 20 proteinogenic amino acids. *Sci. Data*, 3, 2016.

[5] Vito F Palmisano, Carlos Gómez-Rodellar, Hannah Pollak, Gustavo Cárdenas, Ben Corry, Shirin Faraji, and Juan J Nogueira. Binding of azobenzene and p-diaminoazobenzene to the human voltage-gated sodium channel na v 1.4. *Physical Chemistry Chemical Physics*, 23(5):3552–3564, 2021.

[6] Gustavo Cárdenas and Juan J Nogueira. An algorithm to correct for the casscf active space in multiscale qm/mm calculations based on geometry ensembles. *International Journal of Quantum Chemistry*, 121(6):e26533, 2021.