Exercise 3: Calculating Redox Potentials using QM/MM Methods

Gustavo Cardenas
Online Course on Simulation of Biological Systems

February 21, 2022

1 Introduction

In today's exercise we will be computing the redox potential ($\Delta E_{O\rightarrow r}$, or simply ΔE_{red})of the noradrenaline molecule, which was the ligand studied during the first computer exercise held by Vito. Notice that ΔE_{red} is referred to the reduction process (O = oxidized, R = reduced), although we are dealing with an oxidation reaction. This is the convention used in the literature (and in almost every modern chemistry book), so that we will study the energetics of the process:

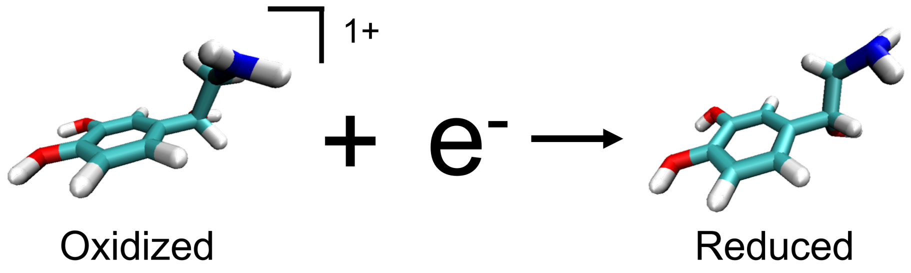

Figure 1: Opposite process of the oxidation of noradrenaline

The redox potential is related to the Gibbs free energy of reaction through the Nernst equation:

$$\Delta G_{red} = -nF\Delta E_{red} \tag{1}$$

where n is the number of electrons involved in the reaction (one in our case) and F is Faraday's constant. Here we will compute ΔG_{red} in electronvolts (eV), so in practice ΔG_{red} will be equal to ΔE_{red}. Of course, the computation of ΔG_{red} will require some sort of sampling, and an important feature of the process we are studying (basically a chemical reaction), we will require sampling on **both the oxidized and the reduced states of the system**. Thus, we will compute the ΔG_{red} as a difference between the energy averages of the reduced state and the oxidized state:

$$\Delta G_{red} =< E_R > - < E_O > -[\Delta G_{e,solv}] \tag{2}$$

Here we have included the free energy of solvation of the electron ($\Delta G_{e,solv}$ = -0.867 kcal/mol), and the brackets indicate that its inclusion is a circumstantial event. If we want to compute (absolute) redox potentials of half reactions (like the one in []), we have to include $\Delta G_{e,solv}$. However, if we compute a, say, total redox process in which we have a redox couple (most often when we consider a reference electrode), the $\Delta G_{e,solv}$ term need not be included, as it is accounted for by both half reactions and thus cancels out. As you may have inferred, $< E_R >$ and $< E_O >$ are energy averages obtained from two MD simulations, each performed on the state of interest. This implies that we will have to construct two different topology+coordinate files and perform two different siulations. In the case of the reduced state (R) we will use the gaff force field and thus, the building-up of the system will be analogous to exercise 1. However, in the case of the oxidized state (O), we will need to slightly modify the force field parameters to account for the fact that, well, it is oxidized. In out case, we will only modify the charge (and implicitly the spin multiplicity - what will be its value in our case?) and use same bonding and van der Waals parameters as in the gaff force field.

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_qmmm_md): Redox potentials via QM/MM MD using the amber/orca-xtb interface.
- Part 2 (2_qmmm_sp): Redox potentials via Single Point QM/MM on MD simulations using classical force fields (MoBioTools)

In this exercise we will use the AmberTools19 package, the Orca quantum chemistry program (in particular, its interface with the xtb software; xtb can be downloaded from xtb) and the MoBioTools toolkit to generate the input files.

At this point, please copy the 3_REDOX folder on your working directory (or otherwise follow the folder architecture in 2), then go to your version of the 3_REDOX folder.

```
cd 3_REDOX
```

This is where the tutorial begins!

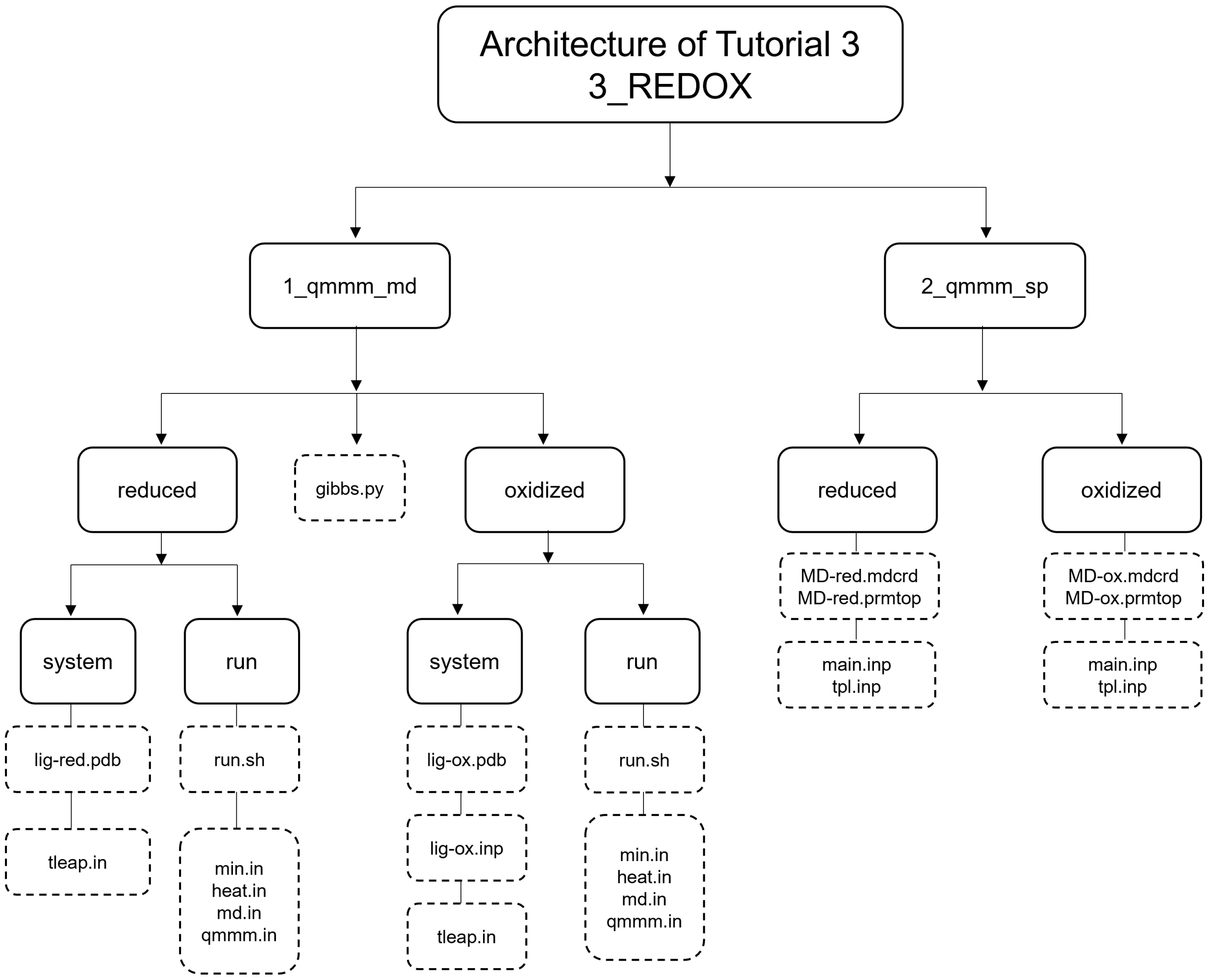

Figure 2: Architecture of exercise 3

2.2 Part 1: Redox Potentials via QM/MM MD

We will start by setting up our systems for the QM/MM MD simulations. Remember that we say "systems" as we will require both the oxidized and the reduced states.

2.2.1 Reduced Species

Please go to the 1_qmmm_md folder:

```
cd 1_qmmm_md
```

There you will find the folders `reduced` and `oxidized`. Each of these folders contains a `system` and a `run` folders. We will have to build two parameter and two coordinate files, so this procedure might result slightly repetitive, but we promise that there will be new stuff to do. Let us begin with the reduced system, which is the more straightforward one. Please go to the `reduced` folder, and then to `system`:

```
cd /reduced/system
```

There you will find (otherwise please get it from our website) the file lig-red.pdb. This is the exact same ligand file that we used on the first tutorial (apart from the atom names), and is precisely the structure of noradrenaline in the Ground State (which here we are calling the reduced state). We will construct the topology and the coordinate files from scratch using tleap, which you probably are acquainted with. First, let us generate a mol2 file containing the point charges. We will use antechamber to do so:

```
antechamber -i lig-red.pdb -fi pdb -o lig-red.mol2 -fo mol2
            -at gaff -c bcc
```

This will generate a mol2 file containing the point charges in the last column to the right of the coordinates. Notice that the atom types correspond to those of the gaff force field, whose parameters we will use for the simulations. Then, use parmchk2 to retrieve missing bonding parameters:

```
parmchk2 -i lig-red.mol2 -o lig-red.frcmod -f mol2
```

Our system will consist of the noradrenaline embedded in a water solvation box. We will use the tleap.in input file to construct the system:

```
source leaprc.gaff
source leaprc.water.tip3p

MOL = loadmol2 lig-red.mol2
loadamberparams lig-red.frcmod

SolvateOct MOL TIP3PBOX 30

saveamberparm MOL MOL-red.prmtop MOL-red.inpcrd
savepdb MOL MOL-red.pdb
quit
```

We can visualize the structure generated to see that everything was done properly, although there's nothing special about it:

```
vmd MOL-res.pdb
```

At this point, we are ready to submit the QM/MM MD simulation. Please copy the MOL-red.prmtop and the MOL-red.inpcrd files to the `run` folder, then go there:

```
cp MOL-red.prmtop MOL-red.inpcrd ../run/.
cd ../run
```

We will perform a minimization, a heating and a production (hereby just called md) using a **classical** force field to bring the system to out working temperature before performing the QM/MM MD simulation. The inputs are the same as in the first tutorial but we will put them here for completeness:

Minimization

```
min.in

Minimization
&cntrl
imin=1          ! Turn on minimization
maxcyc = 200    ! Maximum number of minimization cycles
ncyc = 100      ! 100 steepest-descent steps
ntpr = 2        ! Print energies every 2 steps
ntxo = 2        ! Print binary file
/
```

Heating

```
heat.in

Heat
&cntrl
imin=0,             ! Turn off minimization
ntx=1,              ! Our starting file has no input velocities
irest=0,            ! This is NOT a restart of an old MD simulation
nstlim=3000,
dt=0.002,


ntf   = 2,          ! No Bond interactions involving H
ntc   = 2,          ! Enable SHAKE to constraint all bonds with H


tempi=0.0,          !Initial thermostat temperature in K
temp0=300.0,        !Final thermostat temperature in K
```

```
  ntpr=10,          ! Print energies every 10 steps
  ntwx=100          ! Print coordinates every 100 steps to the trajectory
  ioutfm=0,
  cut=8.0,          ! Nonbonded cutoff, in Angstroms
  iwrap=1,

  ntb=1,            !PBCs in NVT
  ntp=0,            !No pressure control
  ntt=3,            !Temperature control with Langevin thermostat
  gamma_ln=2.0,     !Langevin thermostat collision frequency


  nmropt=1,         !NMR restraints and weight changes read
  ig=-1,
 /


&wt type='TEMP0', istep1=0, istep2=2000, value1=0.0, value2=300.0 /
&wt type='TEMP0', istep1=2000, istep2=3000, value1=300.0, value2=300.0 /
&wt type='END' /
```

Production (MD)

```
 md.in
 Production
&cntrl
 imin       = 0        ! No minimization but molecular dynamics
 irest      = 1        ! Restart from heating
 ntx        = 7        ! Coordinates, velocities and box info read

 ntb        = 2        ! PBCs in the NPT ensemble

 ntp        = 1        ! Isotropic scaling of volumen
 barostat   = 1        ! Berendsen barostat
 pres0      = 1.0      ! Pressure in bars
 taup       = 2.0      ! Relaxation time is ps

 cut        = 8.0      ! Cutoff for L-J and real-space Ewald int
 ntc        = 2        ! Enable SHAKE to constraint all bonds with H
 ntf        = 2        ! No Bond interactions involving H

 tempi      = 300.0    ! Final temperature
 ntt        = 3        ! Langevin thermostat
 gamma_ln   = 1.0      ! Collision frequency in ps-1
```

```
  nstlim      = 1000      ! Number of time steps
  dt          = 0.002     ! Time step in ps

  ntpr        = 1         ! Energy is printed every N steps
  ntwx        = 10        ! Trajectory is printed every N steps
  ntwr        = 20        ! Restart file is printed every N steps
  ntxo        = 1         ! ASCII format for final rst file
  ioutfm      = 0         ! ASCII format for trajectory (xyz) file
 /
```

The really interesting part of this step is the QM/MM MD simulation, which we will perform using as the input geometry the last geometry of the classical MD simulation. The input file is shown below:

QM/MM MD

```
  qmmm.in

  QM/MM MD
&cntrl
 imin        = 0         ! No minimization but molecular dynamics
 irest       = 0         ! Simulation is NOT a restart
 ntx         = 1         ! Velocities and box info NOT read

 ntb         = 2         ! PBCs in the NPT ensemble

 ntp         = 1         ! Isotropic scaling of volumen
 barostat    = 1         ! Berendsen barostat
 pres0       = 1.0       ! Pressure in bars
 taup        = 2.0       ! Relaxation time is ps

 cut         = 8.0       ! Cutoff for L-J and real-space Ewald int
 ntc         = 2         ! Enable SHAKE to constraint all bonds with H
 ntf         = 2         ! No Bond interactions involving H

 tempi       = 300.0     ! Final temperature
 ntt         = 3         ! Langevin thermostat
 gamma_ln    = 1.0       ! Collision frequency in ps-1

 nstlim      = 100       ! Number of time steps
 dt          = 0.002     ! Time step in ps

 ntpr        = 1         ! Energy is printed every N steps
 ntwx        = 1         ! Trajectory is printed every N steps
 ntwr        = 1         ! Restart file is printed every N steps
 ntxo        = 1         ! ASCII format for final rst file
 ioutfm      = 0         ! ASCII format for trajectory (xyz) file
 ifqnt = 1                  ! QM/MM
/
&qmmm
 qmmask     = ':1',         !  QM region mask
 qmcharge   = 0,            !  QM region charge
 qmcut      = 8.0,          !  QM region LJ cutoff
 qm_ewald   = 0,
 qm_theory = 'EXTERN', !   Call external program
/
&orc
method = 'XTB',
/
```

The `&cntrl` section is similar to the md part, the only differences are the fact that we are not restarting the trajectory and the number of timesteps. The sections pertinent to the QM/MM simulation are the `&qmmm` and the `&orc`. The former indicates general settings independent of the QM software (or the internal QM methods in amber), apart from the `qm_theory` keyword, which defines either the methodology we want to use or whether we waht to use an external QM software. The `&orc` section calls the amber/orca interface. In our case, the method we are using is `XTB`, which is a semi-empirical tight binding method to perform 'cheap' QM calculations. In this case we do not need to specify a basis set, but if we used other methodologies such as HF, DFT, MP2, CC, etc., we would need to define a specific basis set. We will perform the calculations sequentially, but we will use the script `run.sh` that will do so for us:

```
   run.sh

#!/bin/bash

# Minimization
sander -O -i min.in -o min.out -c MOL-red.inpcrd -p MOL-red.prmtop
        -r min.rst7

# Heating
sander -O -i heat.in -o heat.out -c min.rst7 -p MOL-red.prmtop
        -r heat.rst7 -x heat.mdcrd

# Classical MD
sander -O -i md.in -o md.out -c heat.rst7 -p MOL-red.prmtop
        -r md.rst7 -x md.mdcrd

# QM/MM MD
sander -O -i qmmm.in -o qmmm.out -c md.rst7 -p MOL-red.prmtop
        -r qmmm.rst7 -x qmmm.mdcrd
```

Please run the simulation as follows:

```
nohup ./run.sh &
```

The calculations will take about 10 to 15 minutes so in the meantime we can start setting up the oxidized system.

2.2.2 Oxidized Species

Please go to the `oxidized/system` folder:

```
cd ../../oxidized/system
```

Here again we will have a pdb file, lig-ox.pdb. Notice that this structure will be slightly different from that of the reduced species - it is the optimized species of the oxidized state. The procedure to generate the topology and the coordinate files will be similar to the previous one, with one exception: the point charges. Indeed, we cannot use the charges obtained with antechamber, as they are not suitable to describe this system (+1 charge, multiplicity 2). The usual procedure is to obtain these charges through an Electrostatic Potential (ESP) Fitting from a QM calculation. This can be easily done using antechamber, but unfortunately it does not have an interface with orca. Therefore, we will use the Mulliken charges (a different set of charges obtained from a QM computation) and "do some manual work" including these charges in the mol2 file of the oxidized species.

We will first create a mol2 file **without** point charges:

```
antechamber -i lig-ox.pdb -fi pdb -o lig-ox.mol2 -fo mol2
            -at gaff -c bcc
```

Notice that the last column to the right of the coordinates marks zero point charges. Then, we will run a single point QM calculation (we will use the Hartree-Fock method with the 6-31G* basis set) using the orca software. The input file is the `lig-ox.inp` file, whose contents are shown below:

```
!HF 6-31G*

*xyz 1 2
O   -0.46100   1.05200   4.73200
C   -0.62300   0.69400   3.45800
C   -1.68300   1.20900   2.71900
C    0.28600  -0.23800   2.84900
O    1.26100  -0.66600   3.66300
C    0.11700  -0.62500   1.51300
C   -0.93400  -0.09000   0.79000
C   -1.86000   0.81300   1.39100
C   -2.99600   1.33300   0.58900
O   -3.95500   1.88700   1.43700
C   -2.50600   2.54500  -0.44200
N   -1.71100   2.20800  -1.54600
H    0.30400   0.59100   5.12300
H   -2.38300   1.88200   3.19900
H    1.85000  -1.31100   3.23500
```

```
H    0.80800   -1.32800    1.05700
H   -1.06200   -0.37300   -0.25100
H   -3.40400    0.55200   -0.06900
H   -4.77000    2.06900    0.94200
H   -2.01000    3.27600    0.20000
H   -3.46000    2.95900   -0.78700
H   -0.71500    2.07500   -1.43900
H   -2.12400    1.77100   -2.35800
*
```

Submit the calculation as follows (the directory containing the orca executable should be on you $PATH):

```
nohup orca lig-ox.inp > lig-ox.inp.out &
```

The calculation will take about a minute. Once this is done, open the lig-ox.inp.out and look for the string "MULLIKEN ATOMIC CHARGES AND SPIN POPULATIONS": as it suggests, below are listed the atomic charges and the so-called atomic spin populations of the atoms. We are interested in the atomic charges, which are in the second column of the table shown:

```
--------------------------------------------
MULLIKEN ATOMIC CHARGES AND SPIN POPULATIONS
--------------------------------------------
   0 O :   -0.711246    0.048553
   1 C :    0.460953    0.375767
   2 C :   -0.165695   -0.371396
   3 C :    0.454876    0.453567
   4 O :   -0.725094    0.057899
   5 C :   -0.226309   -0.386019
   6 C :   -0.069197    0.431468
   7 C :    0.038907    0.415877
   8 C :    0.162803   -0.046769
   9 O :   -0.752008    0.008196
  10 C :   -0.159963    0.007800
  11 N :   -0.865072   -0.000470
  12 H :    0.492065   -0.002244
  13 H :    0.266095    0.015296
  14 H :    0.489529   -0.004161
  15 H :    0.253152    0.016196
  16 H :    0.261047   -0.022673
  17 H :    0.196154    0.001956
  18 H :    0.466186    0.000408
  19 H :    0.199257   -0.000279
  20 H :    0.218201    0.000923
  21 H :    0.347759    0.000092
  22 H :    0.367600    0.000015
```

```
Sum of atomic charges          :     1.0000000
Sum of atomic spin populations:     1.0000000
```

Copy the column containing the Mulliken charges from the lig-ox.inp.out and paste it on the previously created lig-ox.mol2 file, replacing with it the column containing zeros. (There is also the lig-charge.mol2 file on the system directory in case the copying procedure does not work out.)

Optional If you are a user of the vi editor, you can do so in the following manner:

1. Open the lig-ox.inp.out file, then look for the desired string
2. Open the lig-ox.mol2 file in the same window using :vs ligand-ox.mol2
3. You have both files opened simultaneously. You can switch file by pressing Ctrl w w (or fn w w if you are a mac user)
4. move the cursor to the column you are interested in
5. press Ctrl+v (Visual Block mode)
6. highlight the column of interest, then press "y" to copy it
7. Move to the other file to where you want to paste it
8. Paste it with "p"

The rest of the procedure will be the same as in the case of the reduced species: forst generate the missing parameters with `parmchk2`

```
parmchk2 -i lig-ox.mol2 -o lig-ox.frcmod -f mol2
```

and finally, use the following tleap input file to generate the topology and the input coordinate files:

```
 source leaprc.gaff
 source leaprc.water.tip3p

MOL = loadmol2 lig-ox.mol2
 loadamberparams lig-ox.frcmod

 SolvateOct MOL TIP3PBOX 30
 addIonsRand MOL Cl- 0

 saveamberparm MOL MOL-ox.prmtop MOL-ox.inpcrd
 savepdb MOL MOL-ox.pdb
 quit
```

Notice that it is basically the same tleap input as in the case of the reduced species, the only change being (apart from the names of the files) the addition of the `addIonsRand MOL Cl- 0`. This command is necessary since, remember that we have a charged species, but we need to neutralize our system to run MD. Thus, we add an extra Cl- ion to our system. Now we can run the MD snd QM/MM MD simulations sequentially. Before proceeding, the inputs for the minimization, equilibration and production are the same as before, the only relevant change will be in the `qmmm.in` input file, in which we will have to define a different charge and a different spin multiplicity for the QM region:

```
 &qmmm
 qmmask     = ':1',        ! QM region mask
 qmcharge   = 1,           ! QM region charge
 spin       = 2,
 qmcut      = 8.0,         ! QM region LJ cutoff
 qm_ewald   = 0,
 qm_theory  = 'EXTERN', !  Call external program
/
```

As before, copy the MOL-ox.prmtop and MOL-ox.inpcrd files on the /oxidized/run folder, then go to that folder:

```
cp MOL-ox.prmtop MOL-ox.inpcrd  ../run/.
```

and execute the `run.sh` script:

```
nohup ./run.sh &
```

And that's it! Now you have all the necessary simulations to compute the redox potential of the oxidation under study.

2.2.3 Redox Potentials

At this point, recall that to compute the redox potential we need the average energy of the oxidized simulation and of the reduced simulation. Since we are in the oxidized/run folder, let us first retrieve the energies of the oxidized species: grep the string "EXTERNESCF" from the `qmmm.out` file:

```
grep "EXTERNESCF" qmmm.out > energies.dat
```

THis will generate the 2-column file energies.dat. Please delete the last two rows, as these correspond to the average and the RMS fluctuations.

Then, go to the /reduced/run folder and generate a similar energies.dat file there. Notice that inn both cases, the energies are reported in **kcal/mol**, so that since we want our Gibbs free energies in eV, we will have to divide the energies by 23.0605.

At this point let's go to the `1_qmmm_md`, and from there we will compute the averages doing a little bit of python scripting (alternatively, you can use any other method you feel comfortable with). All we will need is the **numpy** library which you probably have already installed if you have installed MoBioTools. Otherwise, you can install it via

```
pip install numpy
```

The following script prints out the value of the Gibbs free energy obtained from our simulation, and can be executed via `python3 gibbs.py`

```
import numpy as np

# Define conversion factor between kcal/mol and eV
conv = 1/23.0605

# Define variables containing input files
ox = "oxidized/run/energies.dat"
red = "reduced/run/energies.dat"

# read the energies for the oxidized and for the reduced species
# each set of energies will be contained in one array

Eox = np.loadtxt(ox, usecols = (1))
Ered = np.loadtxt(red, usecols = (1))

# Print either of the arrays

# Compute averages
av_ox = np.average(Eox)
av_red = np.average(Ered)

# Compute Gibbs Free Energy of reduction
# Also multiply by conv
G = conv * (av_red - av_ox)
print("Gibbs free energy of reduction = {} eV".format(G))
```

2.3 Part 2: Redox Potentials from MD + Single Point QM/MM

This part may be shorter, in that we won't have to perform further MD simulations, but instead we will use previously performed classical MD simulations run for 10 ns. Of course, these simulations were run using a classical force field, to the objective of this part will be to fetch some geometries from each of these trajectories and perform a so-called single point QM/MM computation on top of them. How does this methodology differ from what we have done so far?. Let us go now to the folder 2_qmmm_sp. There you will again see that there are two folders: `oxidized` and `reduced`. Each one will contain a topology file and a trajectory containing 100 geometries. Our objective is to generate 100 orca input files, one for each geometry, for each one of the states. As the procedure will be almost exactly the same, we will follow up the process for the reduced species, and the one of the oxidized species is left as an exercise. We will use the MoBioTools software.

Let us now go to the `reduced` folder. Before we generate the inputs, a **very important** precaution needs to be taken: as we will perform single point calculations, **we won't have PBCs**. Thus, it will be necessary to report the entire trajectory to the primitive cell (autoimage). It is also good practice to put what will be the qm mask at the center of the cell. We will do so, as usual, by using cpptraj:

```
parm MD-red.prmtop
trajin MD-red.mdcrd 1 100
autoimage :1
trajout red-im.mdcrd
go
quit
```

Now, the executable of the MoBioTools package is the `main_qminputs.py` script. Let us see if it works out properly by making it print a help message:

```
main_qminputs.py -h
```

It should print the following message:

```
usage: MoBioTools QM/MM input generator [-h] [-i MAINFILE]
[-t TPLFILE]

optional arguments:
  -h, --help     show this help message and exit
  -i MAINFILE    Input file
  -t TPLFILE     Template file
```

The script takes as input two files: a main file and a template file. The main file contains general information about the QM inputs you want to generate, the name of the topology and coordinate files you want to get the geometries from, the geometry indices to consider, the quantum mechanical mask, and others. Below is the main.inp file contained in the reduced folder:

```
main.inp

&main
tpl     = orca
top     = MD-red.prmtop
traj    = red-im.crd
qmmask  = :1
geoms   = 0 99
&end
```

Here, we will generate QM/MM inputs files for 100 geometries, whose indices span from 0 to 99 in the original trajectory. The template file will contain the instructions that are specific to the type of QM calculation we will perform, and also the syntax specific to the desired QM software. Below is the tpl.inp file contained in the reduced folder:

```
tpl.inp

&header
!XTB
&end

&externchg
```

```
&end

&chgspin
0,1
&end
```

The `&header` section will contain what is usually introduced by a "!" in the orca syntax, and includes the method (XTB in out case), the basis set (none in this case), implicit solvation, etc. The `&externscf` section indicates that we also want to consider the point charges surrounding the QM region (this provides flexibility in cases in which you may want sampling, but want to remove the environment), and the `&chgspin` section introduces the charge and the spin multiplicity of the QM region (**beware:** this section will change in the oxidized case!). Now that we have all the ingredients, we can run the script as follows:

```
main_qminputs.py -i main.inp -t tpl.inp > out.dat
```

It will generate 100 folders, each containing an orca input file, for the 100 geometries in the trajectory file. These folders will be named `geomN, N=0..99`. If we enter, for example, the folder geom10, we can see that there are two files: the orca input itself (`geom10.inp`) and the file `charges_geom10.xyz`, which contains the point charges surrounding the QM region. Now let us perform each single point calculation sequentially, to do so we will use a for loop in command line. Fist, go back to the reduced folder (cd ..), then execute the script `submit.sh`:

```
#!/bin/bash

for i in {0..99}
do
    cd geom"$i"
    orca geom"$i".inp > geom"$i".inp.out
    cd ..
done
```

This time, instead of having a single output file for all the QM calculations, we will have one for each geometry. This would represent a huge advantage over running the QM/MM with amber, since we can retrieve all the information the orca output file provides for each of the geometries (by default amber keeps record of only the outputs of the last two timesteps). In our case, we can still retrieve the energies by simply using the grep command. Since each output file will be of the form `geomN.inp.out`, all we have to do is grep the string "FINAL SINGLE POINT ENERGY" from each output file, as follows

```
grep "FINAL SINGLE POINT ENERGY" geom*/geom*.in.out > energies.dat
```

At this point you will have a file containing the energies (`energies.dat`), although the energies of interest will now be in the fourth column (with the indexing starting from zero). If you repeat this while procedure for the oxidized species (in the 2_qmmm_sp/oxidized folder, this will be an exercise for you) wou will obtain another file `energies.dat`. At this point you can copy the script 1_qmmm_md/gibbs.py into the 2_qmmm_sp folder and modify it suitably to obtain the Gibbs free energy.

With this, today's exercise comes to an end, see you next time for more multiscale QM/MM science!