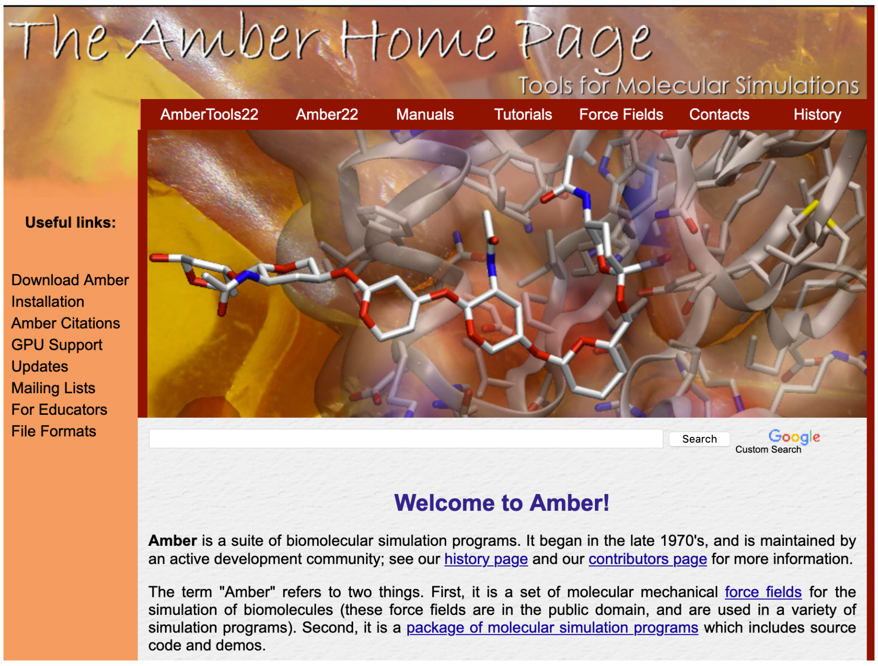
HANDS-ON 1: AMBER, COBRAMM AND QM/MM SYSTEM PREPARATION



Welcome to Amber!

Amber is a suite of biomolecular simulation programs. It began in the late 1970's, and is maintained by an active development community; see our history page and our contributors page for more information.

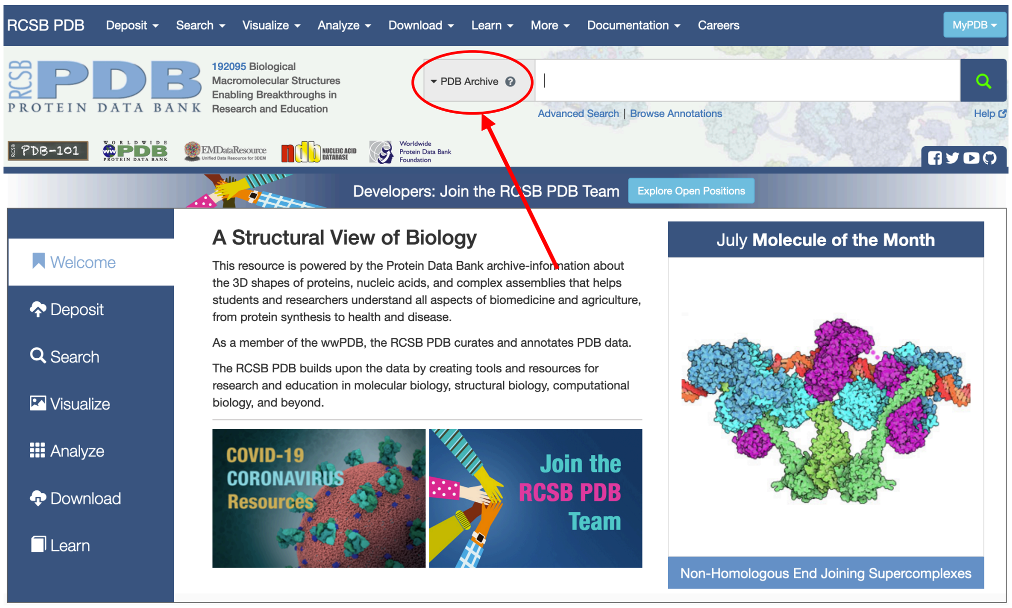
The term "Amber" refers to two things. First, it is a set of molecular mechanical force fields for the simulation of biomolecules (these force fields are in the public domain, and are used in a variety of simulation programs). Second, it is a package of molecular simulation programs which includes source code and demos.

Figure 1 The AMBER home page (https://ambermd.org/)

Amber is both a family of force fields and a software to perform molecular dynamics (MD) simulations. We will use Amber to calculate the classical energy in our excited states QM/MM calculations. For this reason, we need to build and prepare our complex system in an Amber format, so that COBRAMM can parse the correct files and perform the right MM simulations. In this first hands-on, we will first build the system using Amber programs and format. Then we will run some MD to minimize the initial system we built, heat it at room temperature and equilibrate pressure and volume and preparing it for the production run.

PDB preparation

First of all, we need to start from a molecular description of the system we want to investigate. We can either start from a xyz file of a chromophore we want to simply solvate, or from some biomolecular structure of which we want to investigate the excited states dynamics of a residue. We will start from this case, and in particular we will use a file in Protein Data Bank (PDB) format, which contains 3D informations of macromolecules. We will use/download a PDB from the RCSB PDB data bank, where we can find a huge collection of PDBs obtained by experimental data.


RCSB PDB
Deposit
Search
Visualize
Analyze
Download
Learn
More
Documentation
Careers
MyPDB
192095 Biological Macromolecular Structures Enabling Breakthroughs in Research and Education
PDB Archive
Advanced Search | Browse Annotations
Help
Developers: Join the RCSB PDB Team
Explore Open Positions
Welcome
Deposit
Search
Visualize
Analyze
Download
Learn
COVID-19 CORONAVIRUS Resources
Join the RCSB PDB Team
July Molecule of the Month
Non-Homologous End Joining Supercomplexes

A Structural View of Biology

This resource is powered by the Protein Data Bank archive-information about the 3D shapes of proteins, nucleic acids, and complex assemblies that helps students and researchers understand all aspects of biomedicine and agriculture, from protein synthesis to health and disease.

As a member of the wwPDB, the RCSB PDB curates and annotates PDB data.

The RCSB PDB builds upon the data by creating tools and resources for research and education in molecular biology, structural biology, computational biology, and beyond.

Figure 2 The RCSB PDB databank homepage (https://www.rcsb.org/)

Copy either 1w0t.pdb or 2af1.pdb PDB file from

```
/projects/academics/cyberwksp21/Students/davagliano/Instructor_material/AMBE
R
```

PDB files can be visualized with the most common molecular visualization tools (pymol, vmd…).

Important! Visualize the PDB and check everything looks fine.

tleap

We will use *tleap* to create the topology and initial coordinate file of your solvated system.
The program is key-sensitive and very stricked in the input format of the PDB to be loaded. For this reason *Amber* offers a program called pdb4amber that can format any pdb in the correct format for *tleap* . Lunch it as:

```
pdb4amber yours.pdb > yours_amber.pdb
```

Now *tleap* can be launched calling

```
tleap
```

In order to build the system and the topology, first of all the correct libraries containing force field definitions need to be loaded:

`source leaprc.DNA.OL15`	load the OL15 parameters for DNA
`source leaprc.protein.ff14SB`	load the ff14SB parameters for proteins
`source leaprc.water.tip3p`	load the TIP3P force field for water
`source leaprc.gaff2`	load the General Amber Force Field

Now the PDB containing the molecular information can be loaded and we can check everything looks good:

```
INPUT = loadpdb yours_amber.pdb
check INPUT
```

we can now solvate our biomolecule with an octahedron of water molecule with a radius of 20 Angstrom. This represents our unit cell. This cell will have infinite replicas during the classical MM simulations, while a finite-size system will be used during the QM/MM dynamics with COBRAMM. We need to ensure neutrality of the system and we will then also add ions for that.

```
solvateOct INPUT TIP3PBOX 20
addIons INPUT Cl- 0
addIons INPUT Na+ 0
```

Now we can save our toplogy file (.top) and obtain the initial coordinates (.crd) in *Amber* format and leave *tleap*

```
saveAmberParm CHR_SOLV first_run.top first_run.crd
quit
```

sander

Now that we have our topology file and our initial coordinates, we have to prepare the system for the production run. The first step is to minimize the system. We will use two cycles with two different algorithms. For the first … cycle we will use the … algorithm and for the next … the … algorithm instead. We will take the last geometry of the minimisazion procedure and heat the system to 300K. In this step, initial velocities are assigned based on Maxwell-Boltzmann distribution. Once we thermalized the system, we will equilibrate pressure of our unit cell.

We will run these three steps separately lunching the program called *sander* three times. What *sander* mandatorily needs to run the simulations are a topology and coordinate file. As a ouput it will give us three files: i) general output file containing the information of the run, ii) a trajectory file (.trj) containing positions and velocities of the system with a given frequency; iii) a restart file (.rst) which contains positions and velocities of the atoms of the system for the last step of the simulation. The program can be called by command line, where also some non-default names for the output files can be parsed.

The input files for the three calculations can be found at

```
/projects/academics/cyberwksp21/Students/davagliano/Instructor_material/AMBE
R
```

We can run the three steps with:

```
sander -i min_inp -o min_out -p first_run.top -c first_run.crd -x min.trj -r
min.rst
```

```
sander -i heat_inp -o heat_out -p first_run.top -c min.crd -x heat.trj -r
heat.rst

sander -i eq_inp -o eq_out -p first_run.top -c heat.rst -x eq.trj -r eq.rst
```

cpptraj

cpptraj (or the analogous *pytraj*) is a powerful program to perform analyses of the trajectories obtained along your simulation. As an example, here we can simply calculate a distance along the heating trajectory,

In order to do we load *cpptraj* and first load the topology file and then the trajectory file

```
parmin first_run.top
reference first_run.crd
autoimage
loadtraj heat.trj
distance name atom1 atom2 out output_name
go
```

DO IT YOURSELF

Chose a PDB from the databank, download it and convert in *Amber* format. Minimize, heat and equilibrate the system. Run a very short production run and complete a meaningful analysis

antechamber

Copy 3ey0.pdb from the same folder as before. This PDB contains a double strand DNA and a ligand intercalated into it. Unfortunately, there are no parameters available for all the organic molecule and we have to obtain ad-hoc parameters for the ligand before setting up the system. We will use GAFF force field with specifically parametrized charges for the specific molecule. We first write the geometry in xyz format, then transform the xyz file and then use the program *antechamber* to obtain the force field modification for our molecule.

```
module load openbabel
obabel -ixzy bips.xyz -omol2 -Obips.mol2"

antechamber -i bips.mol2 -fi mol2 -o bips.mol2 -fo mol2 -c resp -at gaff2 -nc
2 -dr no

parmchk.log -a Y -i bips.mol2 -f mol2 -o bips.frcmod
```

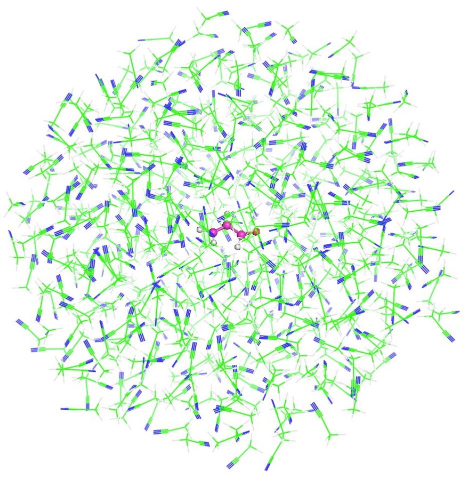
tleap

now we can load tleap and repeat the loading steps as before. Additionally, external set of parameters can be loaded as well:

```
loadamberparams bips.frcmod
```

COBRAMM scripts

As we just saw, setting up the system and preparing the initial snapshot for the QM/MM excited states dynamics can be a long procedure, full of intermediate steps that can go wrong. Additionally, some knowledge of the architecture of *Amber*, of the nomenclature and so on is needed. In order to simplify the this procedure *COBRAMM* offers two alternative way to setup the desired system. In this Hands-on we will see how *COBRAMM* easily allows to solvate a chromophore and prepare the initial conditions for any photochemical calculation can be then performed on such system.



cobramm-solvatedchromo.py

cobramm-equilibration.py

cobramm-droplet.py

cobram.py

cobramm-wignersampling.py

cobramm-post-wigner-equilibration.py

Figure 3 Overview on the auxiliary scripts provided by COBRAMM to solvate a chromophore and prepare the initial conditions for excited states calculations.

TRANS-AZOBENZENE

We will now parametrize, solvate and prepare independently the trans and cis isomer of an azobenzene to study their photophysics.

Copy from

```
/projects/academics/cyberwksp21/Students/davagliano/Instructor_material/COBR
AMM/solvated_preparation
```

the trans and cis azobenzene in xyz format. This will be so far everything you need, then *COBRAMM* will do the rest!

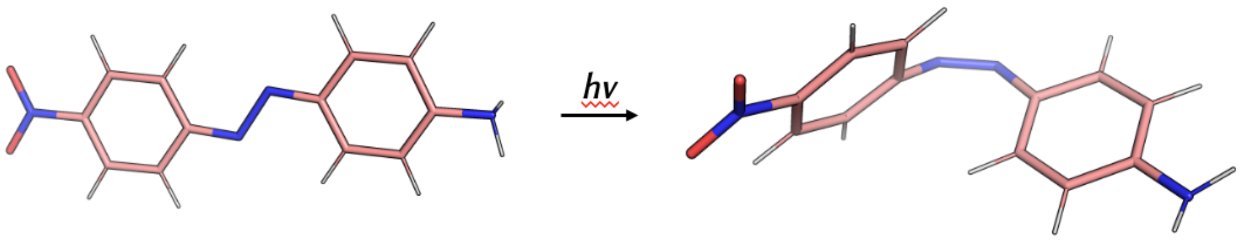


Figure 4 trans-cis isomerization of azobenzene after UV irradiation

We will first solvate the chromophore. Run

```
cobramm-solvatedchromo.py -h
```

then re-run parsing the correct information. After that, to equilibrate the system run

```
cobramm-equilibration.py -h
```

then re-run parsing the correct options. We will now create the finite-size droplet and the input files for *COBRAMM* (cobram.command, real.top, real_layers.xyz). At the stage we are also defined the total radius of droplet and the radius within the solvent molecules will be include in the medium mobile layer.

```
cobramm-droplet.py -h
```

In this way, we solvate and adapted the solvent to the charge distribution of the electronic ground state of the chromophore, In case we are interest to model the solvent response to the electronic distribution of an excited state of the molecule (e.g. if we want to study fluorescence), we have to parse the *cobramm-solvatedchromo.py* the excited states charges. Copy from the same folder S1_charges.out and S2_charges.out and redo the previous procedure by parsing the charges. At the end, check the S_0 and S_{1}-based solvent distributions and find eventual differences.

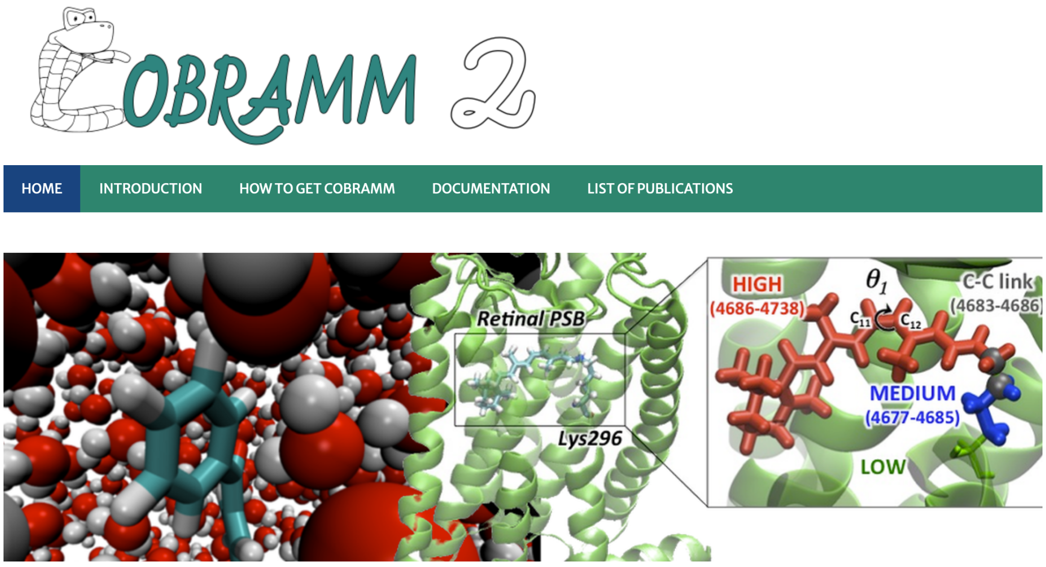
CIS-AZOBENZENE

Redo all the procedure for the cis-azobenzene. Carefully check the structure at the end of the equilibration before continuing!! In case the cis isomer actually comes back to the more stable trans form, but we still want to study it, we need to impose a restrain to keep the molecule still. You can do it yourself, by checking the possible options of *cobramm-equilibration.py*

DO IT YOURSELF

Chose a chromophore of interest, run the COBRAMM script to setup the system in different solvents. Run the scripts with the chromophore both frozen and allowed to move, and run a meaningful analysis to evaluate the different solvent rearrangements.

COBRAMM PREPARE SCRIPT FOR COMPLEX ENVIRONMENT



COBRAMM home page (https://site.unibo.it/cobramm/en)

We already saw how to solvate a molecule, now we will learn how to setup a QM/MM partition with *COBRAMM* when a chromophore of interest is covalently bound to residues we can/want/must include in the MM region. For this task we will use an auxiliary script provided with *COBRAMM* called *Prepare_cobramm_input.pl*. This script reads an input file called *cobram.parm* that the user has to fill with appropriate variables. All the files can be found on:

```
/projects/academics/cyberwksp21/Students/davagliano/Instructor_material/COBR
AMM/Rh_preparation/inputs
```

We will prepare the QM/MM setup to study retinal chromophore embedded in Rhodopsin.

Prepare script

The *Prepare_cobramm_input.pl* solvates the structure given in a PDB file, create the link atom between the chromophore and the neighboring residues, solvate, minimize, heats and eequilibrate the system. If required, it performs a Wigner sampling, post-equilibrating the solvent around each of the geometries sampled and finally prepares the input files for *COBRAMM* run.

The residue of interest, the high-medium-low layers definitions, the link-atom can be all defined by the user in the *cobram.parm*. We can run

```
./Prepare_cobramm_input.pl
```

and check the steps performed printed on the screen. A folder called *cobramm_files* will be created where we can find the real_layers.xyz and the topology files for the whole system (*real.top)* and the H-layer (*model-H.top)* including the the link atom.

Now we can copy the folder

```
/projects/academics/cyberwksp21/Students/davagliano/Instructor_material/COBR
AMM/Rh_calculation
```

and perform some exemplary calculation with *COBRAMM*. We will perform single points at Hartree-Fock and CASSCF level and dynamics on S_1 and nonadiabatic dynamics at CASSCF level. We will change the commands in the *!command* block in the *cobram.command*. Command 1 define the type of calculation. *mdv* indicates dynamic runs. In order to activate Tully fewest switches surface hopping algorithm (FSSH), command *85* must be 1, otherwise adiabatic dynamics will be performed. In order to run optimization, command *1* should say *optgx*, as well as for single point, where in addition command *60* equal to *sp* specify the single point calculation.

Go in the folder *HF-sp* and submit a single point calculation with

```
cobram.py > cobramm.log
```

After that you can go to the CASSCF folders and run a single point, a dynamics on S_1, then modify the cobram.command to activate FSSH algorithm.

We can use *COBRAMM* auxiliary script to analyze the results:

DO IT YOURSELF: chose a PDB from the databank including a chromophore, download it and convert in *Amber* format. Use the *prepare_script*.pl to prepare the system, put the link atom, minimize, heat, equilibrate the system and generate COBRAMM input files. Run CASSCF single point and dynamics on the generated system