Tutorial 1a: Preparing a toy project

Posted on June 18, 2014 by admin

To follow this tutorial a basic knowledge on linux is needed (create, move, copy, edit files and directories).

1) Prepare a PDB file with your system to simulate

Download the following PDB file with a toy 8 aminoacid peptide PDB (pep.pdb). The molecule was built using MOE software, protonated and saved following AMBER naming convention. For a correct preparation of a protein for simualtion, if you are not familiar with the process, please read AMBER tutorials.

2) Create an Amber Object File containing the system to simulate

Open a terminal, create a working directory and move the recently downloaded pdb file there.

```
> mkdir testpymdmix
> cp $DOWNLOAD_DIR/pep.pdb testpymdmix/ # replace $DOWNLOAD_DIR with the p
> cd testpymdmix
```

Once inside, we will create an AMBER object file from the pdb file downloaded. To do so, we will open tLeap (make sure you have installed correctly Amber or AmberTools) using AmberFF99SB:

```
> tleap -f leaprc.ff99SB # or $AMBERHOME/exe/tleap -f leaprc.ff99SB if the
```

Some messages will be printed to screen and finally a new prompt sign (>) will appear. This time we are inside tleap console and ready to load our pdb file:

```
> peptide = loadpdb pep.pdb
Loading PDB file: ./pep.pdb
total atoms in file: 122
> check peptide
Checking 'pep'....
Checking parameters for unit 'pep'.
Checking for bond parameters.
Checking for angle parameters.
Unit is OK.
> saveOff peptide pep.off
> quit
```

We have here 3 commands. The first one (**loadpdb**) will load the molecule contained in **pep.pdb** file and create a tleap unit with the name **peptide**. This unit name is arbitrary and up to your choice. Second command will check that the unit is correctly parameterized (no error in this case). Final command will create a file named **pep.off** where peptide unit will be stored. Finally we exit tleap and we are back to our working directory and with a normal shell prompt.

3) Create a pyMDMix configuration file

Write a pyMDMix project template file using the following command:

Search

News & Updates

Welcoming message
October

Beta testers wanted
March 2

Welcoming message
October

Recent Comments

Meta

Log in
Entries RSS
Comments RSS
WordPress.org

```
> mdmix tools projecttemplate -f pep_test_project.cfg
```

This command will write a file named **pep_test_project.cfg** with some default fields to be filled in. Edit the file with your favorite editor and make the following modifications: Under **[SYSTEM]**:

```
NAME = pep
OFF = pep.off
#PDB =      # Comment out
UNAME = peptide # Un-comment and give the unit name contained in pep.off
```

Under **[MDSETTINGS]**:

```
SOLVENTS = WAT, ETA
NANOS = 1  #Un-comment and write a 1 for simulating only 1 ns
RESTR = HA # Un-comment to apply restraints on all non-hydrogen atoms of t
FORCE = 0.01 # Aplly restraints with 0.01 kcal/mol.A^2 force

# Add the following lines
npt_eq_steps= 250000 # 500ps npt equilibration
nvt_prod_steps= 250000 # 500ps per nvt production file
trajfrequency = 2000  # write trajectory snapshots every 4ps (2000 steps x
```

Other options should remain commented (with # symbol in front of the line). In the first section we have defined a system with name **pep** that will use a unit named **peptide** which is saved inside **pep.off** file (from previous step). In the second section we tell how to simulate this system: will use two solvent mixtures (identified by **ETA** and **WAT** names), each solvent will be simulated for 1ns applying positional restraints over all non-hydrogen atoms (**HA**) of the peptide with a force constant of 0.01 kcal/mol.A^2. Residues part of the peptide are automatically identified, you can specify which residues to restraint at **RESTRMASK** field in this same configuration file changing **auto** to the desired residue number range.

The last 3 lines modify the number of steps to run in the equilibration stage (from the default 500000 to 250000) and the number of steps each production job will simulate (250000, 500ps each job run instead of 1ns default). And will write snapshots of the trajectory every 4ps instead of 2ps (default: 1000, here modified to 2000). All these changes are done to decrease the weight of the simulation for the toy example and to speed up the analysis.

Once the file is saved, we are ready to create the project.

EXTRA: To query what are the solvent mixture identifiers, you can issue the following command:

```
> mdmix info solvents
```

5) Create the project

Using the configuration file, we will create a new project named **pep_amber**. With this command, all files needed for simulation will be created.

```
Parsing md settings for replica creation...
Creating project pep_amber
INFO Creating project folder
INFO Writing system reference file: pep_ref.pdb
INFO Creating replicas for system pep...
INFO Solvating pep with solvent mixture ETA
INFO Solvating pep with solvent mixture WAT
INFO Creating folder structure for replica ETA_1
INFO Writing AMBER simulation input files for replica ETA_1 ...
INFO Restrain mask: :1-8 & !@H=
INFO Creating folder structure for replica WAT_1
INFO Writing AMBER simulation input files for replica WAT_1 ...
INFO Restrain mask: :1-8 & !@H=
INFO DONE
DONE
Total execution time: 12.531s
```

By default **leaprc.ff99SB** and **leaprc.gaff** are loaded. If you wish to use a different forcefield modify **EXTRAFF** field in the project configuration file. We will have a new folder under the current working directory with name **pep_amber**. From now on this folder is the Project folder and any mdmix command will be executed inside this folder. So let's move there:

```
> cd pep_amber
```

And retrieve some information about the recently created project:

```
> mdmix info project
==========================================================
 || pyMDMix User Interface ||
 ==========================================================
 || Author: Daniel Alvarez-Garcia ||
 || Version : 0.1 ||
 ==========================================================
------------------------------
PROJECT pep_amber INFO
------------------------------
SYSTEMS:
========
SYSTEM:pep
REPLICAS
========
REPLICA:ETA_1 system:pep_ETA nanos:1 restrMode:HA restrForce:0.010 Min:Fal
REPLICA:WAT_1 system:pep_WAT nanos:1 restrMode:HA restrForce:0.010 Min:Fal

------------------------------
Total execution time: 0.040s
```

This general description will tell us what systems are included in the project, which replicas have been created for what systems and under which conditions these replicas will be simulated (restraining scheme and length in nanoseconds). Moreover for each of the replicas, a True False flag will indicate the current status of simulation (Min: True – minimization finished -; Eq:True – equilibration process finished -; Prod:True – Production stage finished -; Align:True – trajectory is aligned-). 6) It is time to run the simulation. Jump to Tutorial 1b for instructions on how to submit the simulation and prepare job control files for your own cluster (if needed). Jump directly to Tutorial 1c to download the results and directly proceed with the analysis.

6) Simulation time

It is time to grab all the files and run the simulation. For a general explanation on how to submit the simulation process, go to Tutorial 1b. Skip the simulation step and head to analysis section in Tutorial

1c for quick testing and learning pyMDMix operations (there you can download simulation results to proceed with their analysis).

Posted in Tutorial and tagged tutorial.

Tutorial 1b: Simulating the toy project →