Tutorial 1b: Simulating the toy project

Posted on June 18, 2014 by admin

1) The COMMANDS.sh file

We have our project folder **pep_amber** created in Tutorial 1a ready for simulation. It contains 2 folders inside **MD** directory: **ETA_1** and **WAT_1**. Each one contains all files needed to run an independent simulation: system topology, coordinates, **min/ eq/** and **md/** folders with the configuration files for each simulation step and, last but not least, a **COMMANDS.sh** file.

This file contains all the commands the user should execute to run all the simulation steps. E.g. entering in ETA_1 folder, COMMANDS.sh file contains:

```
cd min
sander -O -i min.in -o min.out -p ../pep_ETA_ETA_1.prmtop -c ../pep_ETA_ETA_1.prmcrd -r min.rst -ref ../p
cd ../eq
pmemd.cuda -O -i eq1.in -o eq1.out -p ../pep_ETA_ETA_1.prmtop -c ../min/min.rst -r eq1.rst -x eq1.nc -re
sh ../check_com.sh eq1.rst
pmemd.cuda -O -i eq2.in -o eq2.out -p ../pep_ETA_ETA_1.prmtop -c eq1.rst -r eq2.rst -x eq2.nc -ref ../pe
sh ../check_com.sh eq2.rst
pmemd.cuda -O -i eq3.in -o eq3.out -p ../pep_ETA_ETA_1.prmtop -c eq2.rst -r eq3.rst -x eq3.nc -ref ../pe
sh ../check_com.sh eq3.rst
pmemd.cuda -O -i eq4.in -o eq4.out -p ../pep_ETA_ETA_1.prmtop -c eq3.rst -r eq4.rst -x eq4.nc -ref ../pe
sh ../check_com.sh eq4.rst
pmemd.cuda -O -i eq5.in -o eq5.out -p ../pep_ETA_ETA_1.prmtop -c eq4.rst -r eq5.rst -x eq5.nc -ref ../pe
sh ../check_com.sh eq5.rst
cd ../md
pmemd.cuda -O -i md.in -o md1.out -p ../pep_ETA_ETA_1.prmtop -c ../eq/eq5.rst -r md1.rst -x md1.nc -ref
sh ../check_com.sh md1.rst
pmemd.cuda -O -i md.in -o md2.out -p ../pep_ETA_ETA_1.prmtop -c md1.rst -r md2.rst -x md2.nc -ref ../pep
sh ../check_com.sh md2.rst
```

Let me explain the file:

- Minimization, Equilibration and Production are stored in separate folders so to run each of the commands, the shell script moves to the appropriate directory.
- By default, minimization step command is constructed to run with **sander** and all other steps with **pmem.cuda**. You can change this default (see bottom of this post). TODO!!!
- **Output file names should always be respected**. During analysis process, the program will try to locate these files in their correct folders.
- In this case, we are running a simulation with harmonic restraints over all non-hydrogen atoms of the peptide. After each simulation step is finished, an extra script called **check_com.sh** is called to run a re-centering, alignment and imaging process of the last step of the simulation (the restart file that will be used by next command). If this step is not correctly done, the simulation might crash or produce artifacts. Internally, this script calls **ptraj** progam.

2) Local simulation submission

You should directly execute this file if you wish to submit all the simulation process at the local machine:

```
> sh COMMANDS.sh >& COMMANDS.log &
```

This will submit ETA_1 simulation to run. You should repeat this process in all the folders inside MD if you want to submit all of them at the same time (ETA_1 and WAT_1 in this case).

3) Preparing a submission to a computing cluster through a queueing system

Usually, you will submit the simulation process in a computing cluster where all jobs are controlled through a GRID system (e.g. Sun Grid Engine, SLURM, etc.) or, colloquially, a queueing system.

In this case, you will need one queue input file for each independent job to be executed. pyMDMix incorporates a tool to help you in the process of setting up the runs. The program will take a template file corresponding to the queueing system you use and fill the blanks in with the corresponding command for each job execution. Read more in Queue Inputs documentation for setting up your own queue template file.

pyMDMix comes with a simple template file for SGE system. We will use it for this example.

```
> mdmix queue list
 ==========================================================
 || pyMDMix User Interface ||
 ==========================================================
 || Author: Daniel Alvarez-Garcia ||
 || Version : 0.1 ||
 ==========================================================
Installed queue system templates: SGE
Total execution time: 0.019s
```

The template should appear in the list. If not, check that the file name and location is correct (read documentation).

```
> mdmix queue write -n SGE
 ==========================================================
 || pyMDMix User Interface ||
 ==========================================================
 || Author: Daniel Alvarez-Garcia ||
 || Version : 0.1 ||
 ==========================================================

INFO Writing Queue SGE input files for replica ETA_1
INFO Writing Queue SGE input files for replica WAT_1
Total execution time: 0.212s
```

This command should be executed inside the project folder (**pep_amber/**). It has created one queue file for each simulation job. E.g. check in **pep_amber/MD/ETA_1/min** folder. You will have now a file **min.in** and a file **min.q**. **min.q** is the queue input file.

All queue files are connected and you will need to submit the minimization and the rest will automatically follow until the last step of the production stage. E.g.:

```
> cd MD/ETA_1/min
> qsub min.q
```

Last lines of min.q are

```
cd ../eq
qsub eq.q
```

Thus when minimization is finished, the queue will change de directory to **../eq** and submit equilibration process automatically.

Remember you should submit min.q in every independent simulation you are running. You might use the following bash loop from within the project folder (pep_amber):

```
for d in MD/*; do cd $d/min; qsub min.q; cd -;done
```

4) Finish MD and move files

If you were running the simulation in a cluster and analyzing in a different location, when all simulations are finished, make sure you copy the resulting files to the appropriate directories. Please keep the same folder structure and naming for the analysis process to work correctly.

Jump to Tutorial 1c to start the analysis.

Posted in Tutorial and tagged tutorial.

← Tutorial 1a: Preparing a toy project

Tutorial 1c: Analyzing the toy project →