GROMACS Tutorial

Computational Photodynamics Research Laboratory

Last edited: 2nd Nov 2017

GROMACS(GROningen MAchine for Chemical Simulations) 는 네덜란드의 Groningen University에서 처음 개발되어 널리 사용되고 있는 분자 동역학(Molecular Dynamics: MD) 프로그램 ‘패키지’이다. GROMACS의 장점은 소스코드가 공개되어 있는 프리웨어이기 때문에 자기에게 맞는 기능을 추가해 넣을 수 있고, 조작이 비교적 쉽고, 계산 속도가 빠르다는 것이다.

1. GROMACS 설치

현재 GROMACS의 최신 버전은 2017년 9월 15일에 release된 2016.4이지만 버전업을 하면서 설치 과정이 대대적으로 개편되어 겪는 어려움 때문에 현재 연구실에서 주로 사용하고 있는 버전은 4.5.5이다.

1-1. Configuration

본격적인 컴파일에 앞서 컴파일러와 라이브러리의 위치를 지정하고, install directory를 비롯한 여러 option을 정해 주고, 원활한 컴파일을 위해 여러 가지 테스트를 하는 등의 과정이다. 이 과정은 Linux에 대한 이해가 어느 정도 필요하므로, 실행은 사수에게 부탁하도록 하자. 아래에서는 몇 가지 option의 의미를 알아보기로 한다.

--disable-float - 부동소수점 연산에서 single precision 대신 double precision을 사용한다.

--program-suffix=\_d - double precision으로 동작하는 프로그램이라는 의미에서, 컴파일된 실행파일의 뒤에 일괄적으로 \_d를 붙인다. 이는 다른 옵션으로 컴파일된 프로그램들이 섞일 경우 서로 구별해 주기 위함이다.

--prefix=<directory> - 실행파일들이 실제로 위치하게 될 장소를 지정한다. 지정하지 않으면, 기본적으로 /usr/local 에 설치하려고 시도하게 되는데, 이곳은 관리자 권한이 없으면 쓰지 못하므로 컴파일이 불가능하다. 따라서 home directory 밑의 적당한 장소를 지정한다.

--enable-mpi - 멀티코어 가동이 가능하게 컴파일한다. (여기서는 사용하지 않음)

1-2. Compile

Configuration을 성공적으로 수행하였다면,

make

를 입력한 다음

make install

을 실행해 주면 컴파일이 자동으로 완료되고 configuration 과정에서 지정해 준 directory에 프로그램들과 관련 파일들이 설치된다.

2. GROMACS를 이용한 분자 동역학 시뮬레이션의 기초

GROMACS가 설치된 directory 하위의 /bin으로 들어가 설치된 프로그램들을 확인해 보자. 각각의 binary들은 하나의 독립된 프로그램이다. 프로그램 실행 커맨드 뒤에 -h 옵션을 주고 실행하면 (예: grompp\_d -h, mdrun\_d -h) 각각의 프로그램들이 무슨 기능을 하는지에 대한 매뉴얼을 불러올 수 있다. 기본적으로 GROMACS manual에 그대로 실려 있는 내용들로, 상당히 자세하게 설명되어 있으므로 프로그램을 실행하기 전에 한 번쯤 읽어 보면 도움이 될 것이다. 이제 ~/.cshrc 의 PATH 부분에 GROMACS 프로그램들이 위치한 directory를 추가해 어디서나 실행이 가능하게 만들자 (사수에게 부탁하자).

MD 시뮬레이션을 사용하여 다루는 대표적인 시스템은 용매화된 단백질 분자이다. 용매 안에서 단백질을 시뮬레이션하기 위한 단계별 과정과 각 단계에 사용되는 프로그램들이 그림 1에 나타내어져 있다. 앞에서도 설명했듯이 각각의 단계에 서로 다른 프로그램들이 사용되는 것을 볼 수 있다. 여기에 대한 세부사항은 실제로 5절에서 시뮬레이션을 연습할 때 따로 설명하기로 한다.



그림 1. 단백질을 용매 속에서 시뮬레이션하기 위한 flow chart

(출처: GROMACS 홈페이지)

보통 MD 시뮬레이션은 다음과 같은 순서로 이루어진다.

- System을 구성

- Energy minimization

- Solvation

- Energy minimization of solvated system

- Heating and equilibration

- Production and data analysis

보통 NVT 시뮬레이션(Canonical ensemble)을 할 때는 system에 용매 효과를 더해 준 후 thermostat을 통한 temperature coupling을 한다. 이 때 먼저 짧은 시뮬레이션을 돌려 equilibration을 해 준 다음 production을 위한 긴 시뮬레이션을 돌리게 된다. Heating과 production을 분리하는 대신 MD를 수행해서 얻어지는 분자의 움직임 데이터(trajectory)의 앞부분을 일정 분량 잘라내는 것으로 equilibration을 대신할 수도 있다.

3. GROMACS에서 사용되는 파일들에 대한 설명

GROMACS에서 사용되는 파일들에는 상당히 많은 종류의 (생소한) 확장자가 붙어 있기 때문에 GROMACS를 처음 접하는 사용자들에게는 혼란스러울 수도 있다. 이 절에서는 주로 사용되는 파일들의 기능을 확장자별로 설명한다. 완벽한 이해를 위해서는 force field의 구조를 비롯해 많은 세부사항들을 알아야 하므로, 대략적인 내용들만 알아보자.

- pdb 파일

pdb 파일에는 단백질에 대한 여러 정보들이 담겨 있다. 그 중 가장 중요한 것은 단백질을 구성하고 있는 원자들의 좌표와 amino acid residue의 배열 순서 등이다. 거의 모든 MD 시뮬레이션 program들이 이를 바탕으로 정보를 구성하며, 향후 실제 시뮬레이션을 수행할 때도 제일 먼저 pdb 파일을 GROMACS가 처리하는 것이 가능한 format으로 바꾸는 일을 할 것이다. RCSB protein data bank ([www.rcsb.org](http://www.rcsb.org)) 에서 다양한 단백질들의 pdb 파일을 검색하고 내려받을 수 있다.

- gro 파일

GROMACS는 pdb2gmx라는 프로그램을 통해 pdb 파일을 gro 파일과 top 파일이라는 두 개의 독립된 파일들로 분리한다. 이 중 gro 파일에는 system을 구성하는 원자들의 좌표가 들어 있으며, 그 단위는 nanometer이다. gro 파일은 pdb 파일을 가공해서만 만들 수 있는 것은 아니며, 시뮬레이션의 결과물로 얻거나 직접 제작할 수도 있다.

- top 파일

top 파일에는 분자를 구성하고 있는 원자들 자체에 대한 정보와 connectivity, 즉 어떤 분자들끼리 bond, angle, dihedral로 연결되어 있는지에 대한 정보가 들어 있다. GROMACS는 이를 바탕으로 force field와 system을 연결시키고 시뮬레이션을 할 수 있도록 만들어 준다.

- tpr 파일

tpr 파일은 gro 파일과 top 파일을 통해 구성한 분자의 interaction parameter (결합/굽힘 힘상수, 전하, Lennard-Jones coefficient 등) 와 시뮬레이션에 사용될 여러 option들 (밑의 mdp 파일을 참조)을 합쳐서 하나의 파일로 재구성한 것이다. tpr 파일은 기계어 (binary) 이며, 이 하나의 파일만 있으면 별도의 input 없이 MD 시뮬레이션을 수행할 수 있다.

- mdp 파일

시뮬레이션에 관한 여러 가지 변수들이 들어 있는 파일이다. 대표적인 것들은 시뮬레이션의 길이, 시뮬레이션의 종류, 데이터의 기록 빈도, time step 등이다. 변수들에 대해서는 밑에 다시 설명하기로 한다.

- cpt 파일

cpt는 checkpoint의 줄임말로, 중단되었거나 종료된 시뮬레이션을 끝난 지점에서 다시 시작하고 싶을 때 사용하는 파일이다.

- edr, trr 파일

MD 시뮬레이션의 결과물로 생성되는 binary이다. edr 파일에는 system의 에너지와 관련된 정보, 그리고 trr 파일에는 system의 trajectory, 즉 각각의 순간에 시스템을 구성하고 있는 원자들의 좌표와 속도에 관한 정보가 들어 있다. 시뮬레이션을 진행하는 도중 일정 시간마다 system의 정보가 이들 파일에 기록되는데, 얼마나 자주 기록할지에 대한 옵션은 앞서 소개했던 mdp 파일에서 설정한다. 사실상 가장 중요한 정보가 담긴 파일들이라고 할 수 있다.

4. 연습 1 - Maxwell-Boltzmann 분포 재현하기

이 절에서는 가장 기초적인 예로써, 입자 하나에 대하여 300 K에서 NVT-constant 시뮬레이션을 수행해 ergodic theorem을 기반으로 물리화학 시간에 배운 Maxwell-Boltzmann 분포를 실제로 재현해 볼 것이다.

gmxexercise.tar 파일의 압축을 풀면 thermostat, protein, coumarin이라는 세 개의 폴더가 보일 것이다. 이 중 thermostat 폴더에 들어간다. 폴더 내부의 Par.top와 Par.gro는 헬륨 원자 하나로 구성된 시스템을 표현하는 파일들이다. 3절에서 설명했듯이, top file에는 원자의 종류와 질량, 전하와 같은 값들이 적혀 있고, gro file에는 원자의 초기 좌표와 속도가 적혀 있다. 이렇게 정의된 system을 충분히 긴 시간 동안 시뮬레이션하고, 헬륨 원자의 kinetic energy를 histogram으로 만들어 이론적으로 구한 distribution과 일치하는지 확인해 본다.

우선 mdp 파일을 열어, 여러 옵션들 중 시뮬레이션의 길이와 데이터의 기록 빈도를 먼저 확인한다. 다양한 parameter들이 있는데, 중요한 것들을 아래에 표시하였다. 우리의 simulation input에는 나타나 있지 않은 것들도 있는데, 이 경우 프로그램에 미리 정해져 있는 기본값이 사용된다.

Integrator – 수치적분에 사용되는 알고리즘. Leap-frog, Velocity-verlet 등의 알고리즘이 있다.

dt – 시뮬레이션에 사용되는 time step의 간격. 단위는 ps이다.

steps – 시뮬레이션을 진행할 step의 수이다.

nstx(v,f)out – trr 파일에 x, v, f를 몇 step마다 출력할지 지정한다.

tcoupl – temperature coupling에 사용할 thermostat의 종류. Berendsen, V-rescale 등의 종류가 있다.

tau-t – temperature coupling의 time constant. 온도 조절을 위한 velocity rescaling을 얼마나 자주 할 것인가를 결정한다.

ref-t – system을 어떤 온도로 맞출 것인가를 결정한다.

pcoupl – pressure coupling에 사용할 barostat의 종류. Berendsen, Parrinello-Rahman 등의 종류가 있다.

tau-p, ref-p – tau-t, ref-t와 비슷하다.

epsilon\_r – implicit solvent model에서 solvent의 dielectric constant를 지정한다.

coulombtype – electrostatic interaction의 계산 방식을 정한다.

vdwtype – Van der Waals interaction의 계산 방식을 정한다.

우리는 300 K에서 11 ns 동안 시뮬레이션을 수행하며, 에너지는 매 step마다 기록할 것이다. 에너지를 알고 있다면 입자의 위치와 속도 정보는 불필요하므로 이 둘은 최소한으로 기록한다. 옵션들의 확인이 끝났다면, 다음의 명령어를 실행한다.

grompp\_d -f Par.mdp -c Par.gro -p Par.top -o Par.tpr -maxwarn 1

Center of mass motion을 억제하라는 GROMACS 프로그램의 경고가 나오지만 우리의 목적을 위해서는 억제하지 않는 것이 맞으므로 그대로 진행하기로 한다. -maxwarn 1은 경고 1개를 무시하고 tpr 파일을 작성하게 해 주는 명령어이다.

grompp\_d 프로그램이 top 파일, gro 파일, mdp 파일을 통합해 시뮬레이션의 input 인 tpr 파일을 만들어냈다. 이제 다음의 명령어를 입력하여 mdrun\_d 프로그램을 이용해 실제 시뮬레이션을 수행한다.

mdrun\_d -s Par.tpr -e Par.edr -o Par.trr -c Par\_out.gro -g Par.log

System의 크기가 매우 작기 때문에, 11 ns 정도의 (계산화학자의 기준에서) 굉장히 긴 시뮬레이션을 하는 데도 그렇게 오랜 시간이 걸리지 않는 것을 볼 수 있다. 보통 104 개 정도의 atom으로 구성된 system의 경우, single core로 돌렸을 경우 하루에 몇 ns 정도가 한계이며, multicore 버전으로 프로그램을 컴파일하여 load를 여러 CPU에 분산시키는 방법으로 속도의 향상이 가능하다. 보통 입자들 사이의 non-bonded (coulomb + vdW) interaction 계산이 시뮬레이션 시간의 많은 부분을 차지한다. 이는, bonded interaction의 개수는 보통 atom 수에 비례하여 증가하는 데 반해, non-bonded interaction의 개수는 atom 수의 제곱에 비례하여 증가하는 것에 기인한다.

시뮬레이션이 종료되면 g\_energy\_d 프로그램을 사용하여 edr 파일로부터 kinetic energy를 추출한다.

g\_energy\_d -f Par.edr -dp -o Par.xvg

를 입력하면 어떤 component에 대해 값을 추출할지를 묻는다. Kinetic energy는 4번이므로, 숫자 4를 입력하고 enter를 2회 누르자. 생성된 file인 Par.xvg를 열어 보면 매 frame마다 kinetic energy 값들이 출력된 것을 볼 수 있다.

우선 C 언어로 간단한 코드를 짜서, 데이터의 처음 1 ns에 대해 1 ps마다 에너지 값을 뽑아서 에너지의 변화 양상을 plot해 본다. 맨 처음 system에 존재했던 excess energy가 thermostat에 의해 감소되어 equilibration으로 가는 것을 눈으로 확인해 보자.

이제, 데이터의 production 부분 (1 ns~11 ns) 에 대하여 모든 data를 사용하여 histogram을 만들어 보자. Histogram의 범위는 0~30 kJ/mol, 계급의 크기는 0.02 kJ/mol 로 하고, 이를 상대도수로 바꾸어 실제로 300 K에서 이론적으로 구한 Maxwell-Boltzmann distribution과 일치하는지 최종적으로 확인한다.



그림 2. Simulation을 통해 구한 Maxwell-Boltzmann distribution과 이론값의 비교

만약, tpr 파일을 만들 때 프로그램의 경고를 받아들여 center of mass motion을 억제하면 어떤 일이 발생하겠는가? mdp 파일의 comm-mode 항목을 ‘Linear’ 로 바꾸고 다시 tpr 파일을 만들어 시뮬레이션을 실행해 보자. 이 결과는, 컴퓨터가 출력하는 메시지를 기계적으로 해석해 실행에 옮기기보다는 자기가 어떤 작업을 하려고 하는지 항상 물리적으로 생각하고 행동하는 것이 중요하다는 것을 보여 준다.

5. 연습 2 - 생물학적인 시스템에 대한 MD 시뮬레이션

실제로 단백질 분자에 대한 시뮬레이션을 해 볼 것이다. 전체적으로 다음과 같은 과정을 통해 진행된다.

시뮬레이션 준비

- 단백질 pdb 파일로부터 top 파일과 gro 파일의 생성

진공 (vacuum) 중에서의 시뮬레이션

- Energy minimization

- 진공에서의 MD 시뮬레이션 (50 ps)

- vmd를 사용한 visualization과 analysis

Periodic boundary condition을 이용한 수용액상에서의 시뮬레이션

- System 주위에 solvent box 만들기

- 용매화된 상태에서의 energy minimization

- 용매화된 상태에서의 MD 시뮬레이션 (50 ps)

- vmd를 사용한 visualization

5-1. 시뮬레이션 준비

protein 폴더에 들어가면 두 개의 pdb 파일이 보일 것이다. 우선 2ECH.pdb는 RCSB protein data bank에서 그대로 다운로드 받은 파일이고, 2ECH\_mod.pdb는 그 중 model 6의 데이터만을 별도로 떼어 놓은 것이다. 보통 RCSB에서 바로 내려받은 pdb 파일은 이렇게 여러 가지 구조들이나 잡다한 데이터들을 포함하고 심지어 몇몇 residue들이 빠져 있는 경우도 있기 때문에 많은 경우 수정을 거쳐야 제대로 된 input으로 사용할 수 있다. 이제

pdb2gmx\_d -f 2ECH\_mod.pdb

를 입력하여 pdb 파일로부터 gro 파일과 top 파일을 만들어 보자. 이를 실행하면 GROMACS가 가지고 있는 force field들의 목록과 함께 어떤 force field를 선택할 것인지 묻는 메시지가 나온다.

1: AMBER03 protein, nucleic AMBER94 (Duan et al., J. Comp. Chem. 24, 1999-2012, 2003)

2: AMBER94 force field (Cornell et al., JACS 117, 5179-5197, 1995)

…

우선 force field로는 8번의 CHARMM27 all-atom force field (with CMAP) - version 2.0을 선택하고, water model로는 1번의 TIP3P를 선택한다. top와 gro 파일이 만들어지는데, 아무 옵션도 주지 않았기 때문에 파일들의 이름이 미리 지정된 기본값인 topol.top와 conf.gro로 설정되었다. 파일들의 이름을 지정해 주고 싶을 때는 다음과 같이 입력한다.

pdb2gmx\_d -f 2ECH\_mod.pdb -o 2ECH\_mod.gro -p 2ECH\_mod.top

2절에서도 설명했듯이, pdb2gmx\_d -h를 입력하면 이 옵션들에 대한 정보를 보여준다.

이제 gro 파일과 top 파일을 vi editor로 열어서 어떤 형식으로 되어 있는지 대략적으로 훑어보고, 각각의파일들에 포함된 정보를 원래의 pdb 파일과 비교해 본다. pdb 파일은 atom들 사이의 결합에 대한 정보를 전혀 알려 주지 않기 때문에, top 파일의 [ bond ] 를 시작으로 열거되어 있는 connectivity에 대한 정보들은 force field에 기반하여 작성된 것이다. Force field의 구조에 대해서는 6절에서 조금 더 자세히 살펴볼 것이다.

5-2. 진공(vacuum) 중에서의 시뮬레이션

MD 시뮬레이션으로부터 의미 있는 데이터를 얻기 위해서는 본격적인 시뮬레이션을 시작하기 전에 언제나 energy minimization을 해 주어야 한다. 어떤 force field를 사용하는가에 따라서 system의 에너지 상태가 달라지게 되고 때때로는 local strain이 생겨 향후 production을 위한 시뮬레이션에 영향을 줄 수도 있기 때문이다. Energy minimization을 위해서는 mdrun\_d 프로그램을 짧은 시간 동안 steepest descent algorithm을 이용하여 돌린다. 우선 input 파일인 tpr 파일을 만들어야 한다. minimize.mdp 라는 파일이 있는지 확인하고 vi editor로 열어 보자.

grompp\_d -f minimize.mdp -c 2ECH\_mod.gro -p 2ECH\_mod.top -o 2ECH\_min.tpr

을 입력해 본다. 그러면 프로그램은 error message를 출력하는데, 이는 system이 담긴 box의 크기가 cutoff length에 비해 너무 짧아서 발생하는 것이다. Box의 크기는 gro 파일의 맨 마지막 줄에 지정되어 있다. gro 파일을 열어 이 값들을 10.0 정도로 고치고 다시 명령어를 입력하면 프로그램이 제대로 동작하게 된다. tpr 파일이 제대로 생성되었는지 확인해 본다.

mdrun\_d 프로그램을 통해 energy minimization을 해 보자.

mdrun\_d -s 2ECH\_min.tpr -c 2ECH\_min.gro -e 2ECH\_min.edr

약간의 시간이 지난 후 메시지와 함께 output 파일들이 출력된다. 출력된 gro 파일이 energy minimization을 거치고 난 system의 구조이다. vmd와 같은 visualization 프로그램을 이용하여 minimization을 하기 전과 후의 구조를 비교하는 것도 가능하다.

Minimization의 결과물로 gro 파일 외에도 edr 파일과 trr 파일이 생성되는데, 여기에는 각각 system의 energy와 trajectory(coordinate + velocity)가 저장된다. 이 파일들은 binary이므로 그대로는 읽을 수 없고, g\_energy\_d라는 프로그램을 사용하여 사람이 읽을 수 있는 ASCII data로 변환해야 한다. 이런 과정을 거쳐 시뮬레이션에서 얻어진 data들을 analysis에 사용할 수 있다. 우선 edr 파일에 대해 간단한 분석을 해 보기로 하자.

g\_energy\_d -f 2ECH\_min.edr -dp -o 2ECH\_min.xvg

이 명령어를 실행하면 어떤 에너지 항목에 대하여 data를 추출할지에 대해서 묻게 되는데, ‘Potential’ 을 선택하면 프로그램은 edr 파일에서 potential energy 항목을 추출하여 2ECH\_min.xvg에 저장한다. vi 에디터를 이용하여 앞쪽의 header 부분을 지우고, x축에 step, y축에 potential energy를 plot하여 energy minimization이 되는 것을 눈으로 확인해 보자.

이제 MD 시뮬레이션을 실행할 차례이다. Energy minimization을 할 때와는 다른 parameter를 써야 한다. md.mdp를 열어 minimize.mdp와 비교해 본 후, grompp\_d를 사용하여 tpr 파일을 만든다.

grompp\_d -f md.mdp -c 2ECH\_min.gro -p 2ECH\_mod.top -o 2ECH\_md.tpr

주의할 점이 있다. 분자의 connectivity는 바뀌지 않았으므로 top 파일은 처음 사용했던 파일을 그대로 사용해도 되지만, gro 파일은 minimization을 통해 얻어진 것을 사용해야 한다. 이제 아까와 마찬가지로

mdrun\_d -s 2ECH\_md.tpr -c 2ECH\_md.gro -e 2ECH\_md.edr

을 입력하여 md 시뮬레이션을 할 수 있다. 마찬가지로 결과물로 얻어진 edr 파일을 g\_energy\_d를 사용하여 변환하고 total energy를 시간에 따라 plot한다. 시뮬레이션 초기 단계에서 thermostat이 열적 에너지를 system에 공급하여, 에너지가 증가하는 것을 확인해 보자.

지금까지의 시뮬레이션은 전부 진공 중에서 이루어졌다. 실제로 생체분자는 solvation이 되어 있는 경우가 많기 때문에 대부분 수용액상에서 MD를 수행하게 된다. 이 경우 solvent effect를 고려해 주어야 한다. Solvent의 효과를 시뮬레이션에 넣기 위한 가장 간단한 방법은 dielectric screening coefficient를 설정하는 것이다. 물의 dielectric coefficient가 약 80이므로 mdp 파일에 epsilon\_r = 80이라는 옵션을 추가하여 다시 시뮬레이션을 실행할 수 있다. 이를 통해 얻어진 결과를 진공 중에서의 시뮬레이션 결과와 비교할 수도 있으나, 아마도 의미 있는 차이점은 찾지 못할 것이다.

5-3. Periodic Boundary Condition을 이용한 수용액상에서의 시뮬레이션

MD 시뮬레이션을 용액 상태에서 하기 위해서는 용매와 system과의 상호작용을 시뮬레이션에 포함시켜야 한다. 여기에는 크게 두 가지 방법이 있다. 5-2절의 마지막 부분에서처럼 dielectric한 환경으로 용매를 대신하는 implicit solvent 방법과, 실제로 물 분자를 system 주변에 둘러싼 후 시뮬레이션을 하는 explicit solvent 방법이 있다. 당연히 explicit solvent model이 실제와 가깝기 때문에 화학적으로 더 의미가 있는 시뮬레이션을 할 수 있지만, 그 대가로 계산량이 늘어나게 된다. 이 절에서는 periodic boundary condition을 이용해 system이 담긴 정육면체 형태의 box를 물 분자들로 채우고, 이 상태에서 MD 시뮬레이션을 해 볼 것이다.

5-2절에서 얻어졌던 minimized geometry를 담은 gro 파일을 물 분자로 solvation시켜 보자. 이 경우 genbox\_d 라는 명령어를 사용한다.

genbox\_d -cp 2ECH\_min.gro -cs spc216.gro -o 2ECH\_solv.gro -p 2ECH\_mod.top

여기서 -cp는 용질, 즉 system을 지정해 주는 옵션이고 -cs는 용매를 지정해 주는 옵션이다. 우리는 tip3p water model을 사용하지만 spc216 water model의 gro 파일을 지정해 주었다. 이렇게 해도 되는 이유는 tip3p와 spc216이 같은 3-site model이기 때문에 model이 다르기 때문에 오는 차이를 짧은 시간 동안 equilibration을 통해 제거할 수 있기 때문이다. 만약 더 자세한 4-site model인 tip4p model이나 기타 다른 model을 사용한다면 그에 맞는 gro 파일을 지정해 주어야 할 것이다. 기본 제공되는 용매의 gro 파일들은 GROMACS가 설치된 directory의 /share/gromacs/top에 들어 있다.

이제 vmd로 생성된 2ECH\_solv.gro를 열어 보면, 정육면체형의 solvent box가 생성된 것을 확인할 수 있다. 하지만 system이 solvent box의 안에 위치하지 않고 모서리 부분에서 box 밖으로 튀어나와 있을 것이다. 이런 현상을 해결하기 위해서는 editconf\_d 프로그램을 사용한다. 이 프로그램은 system의 center of mass를 box 내부의 원하는 곳으로 이동시키거나, 평행 및 회전이동을 할 수 있게 해 준다. Box가 10nm 크기의 정육면체이므로 -center 5 5 5라는 옵션을 줌으로써 system의 center of mass를 box의 한가운데 갖다 놓을 수 있다.

editconf\_d -f 2ECH\_min.gro -o 2ECH\_min\_center.gro -center 5 5 5

이제 2ECH\_min\_center.gro를 다시 solvation시켜 주자. 그 전에 2ECH.top을 열어 맨 마지막 줄의 solvent 정보를 지워야 한다. 또는, 앞에서 백업된 #2ECH.top.1#을 사용해도 된다.

genbox\_d -cp 2ECH\_min\_center.gro -cs spc216.gro -o 2ECH\_solv.gro -p 2ECH\_mod.top

얻어진 2ECH\_solv.gro를 vmd로 열어 system이 box 중앙에 잘 들어갔나 확인해 보자.

이제 minimize를 하고 md 시뮬레이션을 수행할 차례이다. tpr 파일을 만들고 시뮬레이션을 하는 것은 앞에서와 똑같은 방법으로 할 수 있다.

grompp\_d -f minimize.mdp -c 2ECH\_solv.gro -p 2ECH\_mod.top -o 2ECH\_solv\_min.tpr

mdrun\_d -s 2ECH\_solv\_min.tpr -c 2ECH\_solv\_min.gro

grompp\_d -f md.mdp -c 2ECH\_solv\_min.gro -p 2ECH\_mod.top -o 2ECH\_solv\_md.tpr

mdrun\_d -s 2ECH\_solv\_md.tpr -c 2ECH\_solv\_md.gro -e 2ECH\_solv\_md.edr -o 2ECH\_solv\_md.trr

단순히 위 command를 copy해서 입력하는 것보다는, 한 단계에서의 output이 다음 단계의 input으로 사용되는 과정과 그 의미를 눈여겨보는 것이 좋다. 상당히 오래 걸리는 작업이므로, 작업이 진행되는 동안에 6절의 내용을 먼저 읽어보도록 하자.

md 시뮬레이션에서 얻어진 trajectory인 2ECH\_solv\_md.trr을 trjconv\_d를 사용하여 gro 파일로 변환한 다음 vmd로 열어 보면 system이 움직이는 과정을 실시간으로 관찰할 수 있다.

5-4. 결론

우리가 수행한 시뮬레이션은 단순히 용매 속에 단백질을 띄워 놓고 짧은 시간 동안 관찰하는 정도이기 때문에 실제로 유용한 정보를 추출해 내지는 못한다. 그러나 이러한 방법을 바탕으로 protein의 radius of gyration을 측정하여 실제 실험 과정에서 단백질의 접힘 상태를 알아 내거나, solvation free energy를 계산하여 ensemble level에서 ligand-protein 결합 에너지를 이론적으로 계산하는 등의 여러 가지 의미 있는 일을 할 수 있으며, 현재 이렇게 수천~수십만 개의 원자를 포함한 복잡한 화학/생물학적 시스템을 다룰 수 있는 계산화학적 방법은 MD 시뮬레이션이 유일하다.

6. Force Field

Force field는 MD 시뮬레이션에 관한 여러 parameter의 집합체이다. 즉 bonded interaction과 non-bonded interaction을 기술하는 하나의 model이라고 할 수도 있다. 같은 system이라도 시뮬레이션에 사용된 force field가 다르다면 별개의 시뮬레이션이 된다. GROMACS에서는 20개 정도 되는 force field를 기본 제공하는데, 여기에 외부에서 가져와서 추가할 수도 있다. 시험삼아 force field를 추가해 보고, 이를 이용해 MD 시뮬레이션을 돌리려는 시도를 해 보자.

California State Univ.의 Eric J. Sorin 교수가 AMBER라는 MD 프로그램의 force field를 GROMACS용으로 고친 버전이 <http://ffamber.cnsm.csulb.edu/#install> 에 업로드 되어 있다. 최신 버전을 다운받아서, GROMACS가 설치된 directory 안의 /share/gromacs/top 에 옮겨 압축을 풀고, ffamber03이라는 directory를 상위 폴더로 옮긴다.

각각의 파일들에 대해 간략한 설명을 하자면

.rtp - residue database. Amino acid나 nucleic acid들의 원자 전하량이나 connectivity에 대한 정보

.itp - interaction database. Bonded/nonbonded interaction들에 대한 정보

.hdb - hydrogen database. pdb 파일에 수소가 없는 경우 어떻게 수소를 추가할지에 대한 정보

.tdb - termini database. Terminal residue를 어떻게 처리할지에 대한 정보

이렇게 옮겨 주어도 GROMACS는 새로 추가된 force field가 있다는 사실을 인식하지 못한다. 이를 위해서는 첫째, directory의 이름이 .ff로 끝나야 하고 둘째, ffamber03 directory 내부에 forcefield.doc와 forcefield.itp라는 두 개의 파일이 있어야 한다. ffamber03 directory의 이름 뒤에 .ff를 붙여 ffamber03.ff로 바꾸고, vi editor로 forcefield.doc을 새로 만들어 첫번째 줄에 Force Field의 이름을 적당히 적어 넣는다. 그리고 ffamber03.itp라는 파일의 이름을 forcefield.itp로 바꿔 주면 된다.

이제 pdb2gmx\_d로 top 파일과 gro 파일을 만들어 보자. 우리가 새로 설치한 force field가 추가된 것을 확인할 수 있을 것이다. 하지만 이 force field를 선택하면 error를 출력할 것이다. Error message를 잘 읽어 보고, 아래 설명한 일련의 절차를 통해 순차적으로 error의 원인을 규명하고 수정해 주어야 한다.

- 맨 처음 나오는 error는, ffamber03 Force Field가 N-terminal과 C-terminal 처리를 제대로 하지 못하는 데서 기인한다. 사실 ffamber03은 terminal residue에 대한 정보들을 따로 가지고 있지만, pdb 파일에는 어떤 residue가 terminal이라고 명시되어 있지 않기 때문에 오류가 발생하게 된다. pdb 파일을 열어 1번 residue인 GLU와 49번 residue인 THR을 다른 이름으로 고쳐 주어야 하는데, 무엇으로 고칠지는 ffamber03.ff directory 안의 ffamber03.rtp를 보고 추측해 보자.

- 이 error를 고치면, Lysine에 해당되는 LYS라는 residue가 rtp database에 존재하지 않는다는 error가 나온다. ffamber03에서 Lysine을 나타내는 residue type은 LYS가 아니고, protonation state를 구별하기 위해 두 가지로 구분된다 (LYN, LYP). 두 residue type 중 pdb 파일의 상태와 일치하는 것은 무엇인지 rtp file과 pdb file을 대조하여 알아내 보자.

- 다음 error는 Histidine에 해당되는 HSE라는 residue에 대한 것이다. Histidine의 경우에도 세 가지 다른 protonation state를 나타내는 3가지 residue type이 있다. 마찬가지로 pdb 파일을 확인하고 고쳐 준다.

- 이제 residue 4번 Serine의 HG1 atom이 rtp 파일과 일치하지 않는다는 error를 해결해야 한다. rtp 파일을 확인해 보고 pdb 파일의 HG1을 rtp 파일의 atom name에 맞게 수정해 준다.

- 마지막 error는 조금 까다로운데, rtp 파일에서 C-terminal인 Threonine의 부분을 직접 찾아, atom OC1을 O로, OC2를 OXT로 수정함으로써 해결할 수 있다.

새로운 force field를 사용하는 과정에서는 위에서와 같이 예상치 못한 error들이 발생할 수 있고, 원인을 찾아서 하나하나 수정해 주는 작업이 때때로 매우 번거로워질 때가 많다. 오류 없이 top file과 gro file을 만드는 데 성공했다면 다음 절로 넘어간다.

7. 연습 3 - Coumarin 153 색소 분자의 solvent response function 관찰

(참고 문헌: *J. Chem. Phys.* **135**, 014107 (2011))

이 절에서는 색소 분자인 Coumarin 153의 바닥 상태와 들뜬 상태 force field를 이용하여, 분자가 들뜬 상태로 전이한 후의 relaxation process를 보여 주는 solvent response function을 구할 것이다.

7-1. 이론적 배경

Stokes shift는 색소 분자의 absorption maximum과 fluorescence maximum의 차이로 정의된다. 보통 분자의 바닥 상태와 들뜬 상태의 에너지 표면 (potential energy surface) 형태는 다르고, 가장 안정한 구조도 다르다. 따라서 색소 분자가 빛을 흡수하기 전에 바닥 상태의 안정한 구조에 놓여 있었다 하더라도 전자 전이가 일어난 들뜬 상태 에너지 표면에서는 불안정해지며, 이 경우 분자는 여분의 진동 에너지를 방출하여 들뜬 상태 하에서의 안정한 분자 구조로 되려는 경향성을 갖는다. 빛을 흡수한 직후에 아무런 시간 지연 없이 형광 방출이 일어난다면, 분자의 구조 변화가 일어나지 않았으므로 (Franck-Condon approximation) Stokes shift는 없을 것이다. 실제로 fluorescence는 nanosecond 정도의 비교적 긴 시간대에서 일어나는 현상이며, 이 정도면 진동 에너지를 방출하고 relaxation이 일어나기에 충분하다. 따라서 Stokes shift의 값은 시간이 지남에 따라 점점 증가하여 충분히 긴 시간이 지나면 한 값으로 수렴하게 된다. 보통 Stokes shift라 하면 long-time limit에서의 값을 지칭하며, 그 크기는 분자마다 다르다.



그림 3. Rhodamine 6G의 Stokes shift

(출처: Wikipedia)

시간에 따른 Stokes shift의 변화를 dynamic Stokes shift라 하는데, 다음과 같이 정의된다.

 

꺾인 괄호 <> 는 ensemble average를 나타낸다. 이를 normalize 하여 초기값을 1로 만든 것을 solvent response function이라 한다.

 

Solvent response function은 time-resolved pump-probe experiment 등을 통하여 실험적으로 관측할 수 있다. 시뮬레이션을 통해 이 값을 얻어낼 수도 있는데, 정확한 결과를 얻기 위해서는 물론 바닥 상태와 들뜬 상태를 기술하는 force field의 정확성이 담보되어야 한다.

7-2. MD 시뮬레이션을 통한 solvent response function의 계산

coumarin 폴더에 들어가 sample.gro 파일을 vmd를 이용하여 열어 보면 methanol 용매 분자들 사이에 solvation 되어 있는 Coumarin 153 분자를 볼 수 있다. 이를 300K, 1기압 하에서 1 ns equilibration 후 1 ns production run을 수행하여, production 부분에서 10 ps마다 frame을 저장한 결과물이 ensemble 폴더 안에 있는 100개의 gro file들이다. excited state를 모델링한 force field로는 excited\_chg.top과 excited\_full.top 두 개가 주어져 있다. 전자는 coumarin molecule을 구성하는 원자들의 전하만 바꾼 것이고, 후자는 결합/굽힘 힘상수 등 bonded interaction parameter까지도 excited state에 맞게 새로 만들어 준 것이다.

이제 각각의 gro file을 순서대로 택하여 다음과 같은 과정을 통해 2 ps 동안의 solvent response function을 구해 보자. 첫 번째 gro file인 COU001.gro를 기준으로 설명한다.

- 우선, 빛을 흡수하여 excited state로 올라간 분자를 나타내기 위해 excited\_chg.top과 COU001.gro를 이용하여 2 ps 시뮬레이션을 돌린다. mdp 파일은 excited.mdp를 사용한다. trr 파일과 edr 파일의 이름은 COU001.trr과 COU001\_e.edr 로 지정한다. 이 시뮬레이션을 통해 5 fs마다 system의 원자들의 위치와 에너지가 저장된다.

- 를 얻기 위해서는 위에서 저장된 snapshot들에 대해서 excitation energy의 시간에 따른 변화 를 구해야 하는데, 그러기 위해서는 excited state의 에너지뿐만이 아니라 ground state의 에너지도 계산해야 한다. 앞 단계에서 system이 거쳐 간 geometry들이 COU001.trr 에 저장되어 있음을 기억하자. tpr 파일을 만드는 데 있어서 ground.top 과 ground.mdp 를 사용하되, MD 시뮬레이션 단계에서 -rerun COU001.trr 라는 옵션을 추가로 주면 프로그램은 COU001.trr 에 저장된 snapshot들을 쫓아가면서 ground state energy 만을 계산하게 된다. trr 파일과 edr 파일의 이름을 COU001\_rerun.trr 과 COU001\_g.edr 로 지정한다. COU001\_rerun.trr은 필요가 없는 데이터이므로 시뮬레이션이 끝난 후 지워도 상관이 없다.

- 이제 COU001\_g.edr 와 COU001\_e.edr 로부터 energy gap의 시간에 따른 변화를 구할 수 있다. 어떤 에너지 항목을 선택해서 추출해야 할지 고민해 보자.

- 100개의 gro file에 대해 위 작업을 반복 수행하고 결과들을 바탕으로 를 구한다. 의 값으로는 -26.8630 kJ/mol 를 이용한다. 이와 같은 반복 작업을 할 때는 작업을 자동으로 수행하는 shell script를 만드는 것이 도움이 된다. srf\_charge directory 내부로 이동하여 run\_srf.sh 를 열어서 내용을 확인하고 실행해 보자. 며칠 정도의 시간이 소요되는 상당히 오래 걸리는 작업이기 때문에 시간 낭비를 하지 않도록 작업 중간중간 진행 상황을 확인하면서 언제 끝날지 예측하고 다음 작업을 준비한다.

- 모든 계산이 끝나면 또 하나의 excited state model인 excited\_full.top 에 대해서도 위 과정을 반복해 를 구해 보자. srf\_full 이라는 이름의 directory를 만들고 앞서 사용했던 script file을 복사한다. 사용하려는 topology file이 다르므로, 여기에 맞게 csh script를 수정하는 작업이 물론 필요하다. 이 경우 의 값은 -363.8034 kJ/mol 이다.

- 최종적으로 얻어진 두 개의 를 비교한다. Charge만 바꾸었을 경우와 bonded interaction parameter까지 바꾸었을 경우에 가장 큰 차이는 무엇인가? 이를 각각의 force field의 성질과 관련지어 설명해 보자.