Tutorial of Peptide Nucleation

By Xuan Tang and Wei Han

Correspondence: hanw@pkusz.edu.cn

Table of Contents

Citation 2

Prerequisites 2

Contents in this package 3

Installation 5

Simulation setup of peptide nucleation of aβ16-21 5

Free energy construction of peptide nucleation of aβ16-21 9

Kinetic transition network building of peptide nucleation of aβ16-21 12

Citation

Please cite the following papers when using the collective variable and kinetic transition networks for peptide nucleation,

1) Xuan. T.; Han, W. J. Phys. Chem. Lett. 2022, 13, 5009–5016.
2) Cai, X.; Han, W. J. Chem. Inf. Model 2022, doi.org/10.1021/acs.jcim.2c00066.
3) Han, W.; Schulten, K. J. Chem. Theory Comput. 2012, 8, 4413.

If possible, please also cite the following paper as PACE is coupled with the MARTINI force field,

4) Marrink, S. J.; Risselada, H. J.; Yemov, S.; Tieleman, D. P.; de Vries, A. H. J. Phys. Chem. B 2007, 111, 7812.

Prerequisites

The following software is required for conducting simulations with enhanced peptide nucleation:

1) GROMACS 4.x is required to prepare necessary files for setting up simulation but once prepared, simulations can be performed with later versions of GROMACS. GROMACS 5.x is highly recommended for actual simulations as this version has gained much improved performance for parallel simulations or by GPU acceleration. The installation guideline of GROMACS is detailed at www.gromacs.org;
2) PLUMED 2.3.7 is required to perform the metadynamics simulation. The installation guideline of PLUMED is detailed at https://www.plumed.org/doc-v2.3/user-doc/html/index.html;
3) FFTW 3.3.8 is needed to support the installation of PLUMED 2.3.7. To install FFTW, please visit www.fftw.org;
4) Python 2.4 or later is needed (of note, Python 3 is not supported currently). To install Python, please visit www.python.org;

5) A C compiler is required. A GNU C compiler is recommended and will be assumed to be the working C compiler throughout this tutorial;
6) VMD, a visualization software, is optional but highly recommended. To install VMD, please visit http://www.ks.uiuc.edu/Research/vmd/.

Contents in this package

1) The *Installation/* directory contains a folder named of *Scripts/*, which contains a script needed to be added in PLUMED source code before PLUMED installation: (Throughout this tutorial, a name of an executable or a script will have grey shade, and command lines to be entered are marked yellow)

   Scripts/ADO.cpp: the C++ source code used to be a collective variable for enhancing sampling for peptide nucleation. Needs to be placed in the *src/* directory of PLUMED source code;
2) The *Simulation Setup/* directory contains two folders named of *Scripts/*, which includes a script needed to build simulations, and *Example/*, which contains five simulation parameter files (*.mdp*) for five different types of simulations and two working examples *case1/* and *case2/* for metadynamics simulation of $A\beta_{16\text{-}21}$ trimer and bias-exchange metadynamics simulation of $A\beta_{16\text{-}21}$ 18-mers, respectively:

   Scripts/run.sh: the script for generating box, adding ions and pre-equilibrium processes. The input for this script needs to be a coordinate file in a GRO format, normally attainable from handling of different sources such as the protein data bank by GROMACS;

   Example/em-posres.mdp: energy minimization with position restrains of heavy atoms using the steepest descent method;

   Example/em mdp: energy minimization using the steepest descent method;

   Example/nvt.mdp: pre-equilibration simulations at constant temperature and in constant volume;

   Example/npt.mdp: pre-equilibration simulations at constant temperature and pressure;

   Example/full.mdp: production simulations at constant temperature and pressure.

   Example/case1/: a working example for metadynamics simulation of $A\beta_{16\text{-}21}$ trimer, containing the initial structure in the GRO format and PDB format, topology files, the production simulation file in tpr format and the metadynamics file with dat suffix;

Example/*case2*/: a working example for metadynamics simulation of A$\beta_{16\text{-}21}$ 18-mers, containing the initial structure in the GRO format and PDB format, topology files, the production file in tpr format and the metadynamics files with dat suffix;

3) The *Free Energy Construction*/ directory contains two folders, one named of *Scripts*/, which includes scripts used to do clustering, calculate the free energy of oligomer size $F(n)$ and the equilibrium probability $P_{eq,n}(n_\beta, m)$ of finding from n-sized clusters those having n_β β-strands and m maximum burial depth of misregistered β-strands and corresponding free energy $F(n, n_\beta, m)$, and another folder named *Example*/, which includes the files as a working examples of calculating free energy:

   Scripts/cal-cluster.py and *Scripts*/networkutil.py: scripts that can be used to do clusters for each frame based on distance between residues;

   Scripts/re-boot_strapping-frame.py and *Scripts*/combSum.py: scripts that can be used to calculate the free energy respect to oligomer size;

   Scripts/cal_buried_rc.py and *Scripts*/find_buied_strand.py: used together to get values of n_β and m for different oligomer sizes;

   Scripts/distribution.py: used to calculate the distribution of $P_{eq,n}(n_\beta, m)$ for different oligomer size n;

   Scripts/get_f.py: calculate the free energy respect to n, n_β and m by $F(n, n_\beta, m) = -kTlog(P_{n,eq}(n_\beta, m)) + F(n)$;

4) The *Kinetic Transition Network*/ directory contains two folders named of *Scripts*/, which includes scripts to build kinetic transition networks and do Monte Carlo simulations, and *Example*/, which contains a working example of the process:

   Scripts/map_network_rate.py and *Scripts*/networkutil.py: scripts used together to build kinetic transition networks, catch the states and their free energies in the network, and calculate the rate constant of each transition between states;

   Scripts/KMC.py: the script used to do Monte Carlo simulations.

Installation

Provided that GROMACS 5.x has been installed with its root directory being at *~ROOT/ gromacs5/*, and the source code of PLUMED 2.3.x has been downloaded and unpacked at *~ROOT/plumed2/*, where "~ROOT" is a placeholder for the actual directory. And also make sure that you have downloaded the package and unpacked it at a working directory at *~WORK/* so that all the associated scripts can be found at *~WORK/Peptide_Nucleation/*, please follow the instructions below step by step to complete the installation of PLUMED with the ADO.cpp:

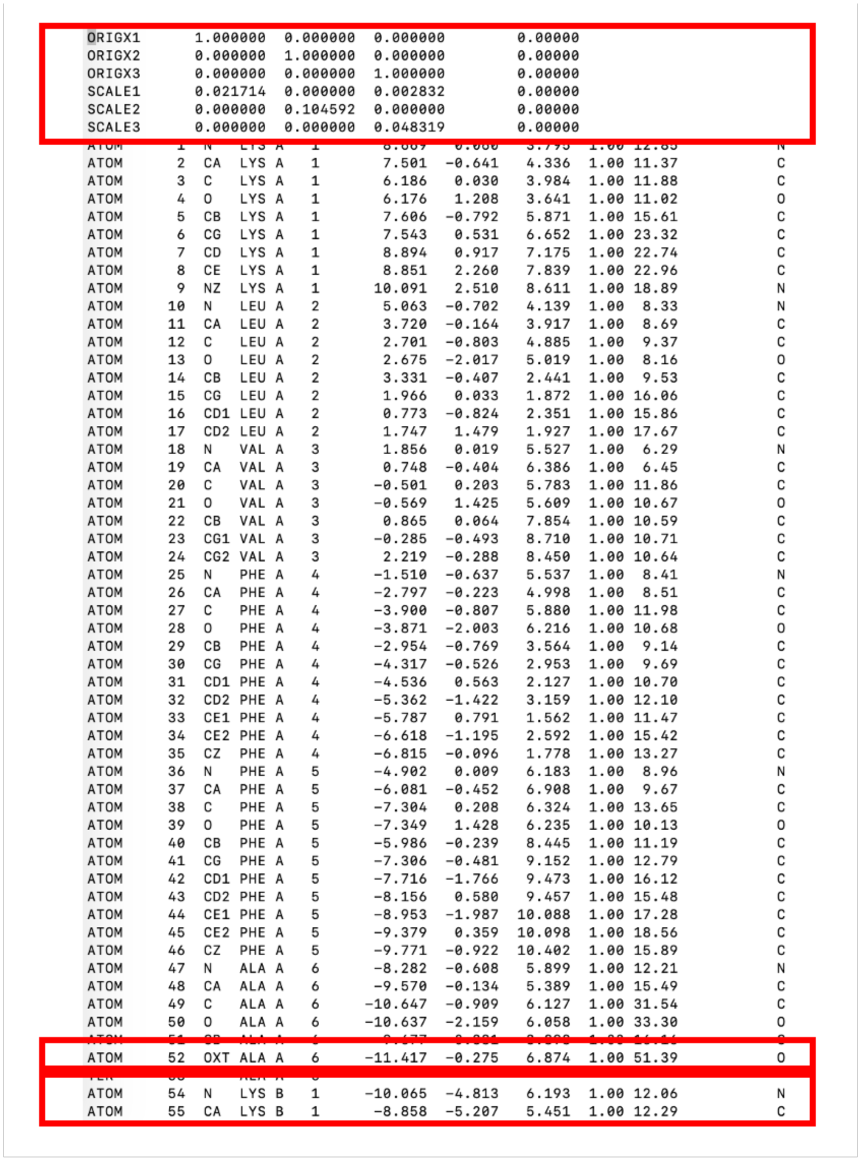
1) Duplicate the *Installation*/*Scripts*/ADO.cpp file in *~WORK/Peptide_Nucleation/* to the folder of plumed source code storing source codes of collective variables (i.e., *~ROOT/plumed2/src/*).
2) The manual of installing PLUMED and patching it into GROMACS as the installation guideline mentioned in its website (https://www.plumed.org/doc-v2.3/user-doc/html/index.html).

Simulation setup of peptide nucleation of Aβ16-21

We will show here a working example of how to set up peptide nucleation metadynamics simulations in the force field of PACE for a short peptide known as Amyloid-β fragment 16-21 (A$\beta_{16\text{-}21}$ for short), which is the core fragment of amyloid-β proteins and has been extensively studied computationally. Let us assume that the current directory contains a copy of all the files from *~WORK/Peptide_Nucleation/Aβ16-21/*. Listed below are the main steps of model setup:

1) The first step is installation of PACE-ASM force field according to https://github.com/hanlab-pkusz/hanlab/tree/master/Tutorial_PACE-ASM. Now we assume the force filed files and associated scripts have been downloaded, unpacked and placed in appropriate directories, and then PACE-ASM force field and GROMACS 4.x (at *~ROOT/ gromacs4/*) have been installed following the manual in the website.
2) Next, we prepare the PDB file 3OW9.pdb, downloaded from PDB library

(www.rcsb.org), to be the initial structure. Since origin PDB file has two chains of A$\beta_{16\text{-}21}$ and some irrelevant information, shown as figure below,



```
ORIGX1      1.000000  0.000000  0.000000        0.00000
ORIGX2      0.000000  1.000000  0.000000        0.00000
ORIGX3      0.000000  0.000000  1.000000        0.00000
SCALE1      0.021714  0.000000  0.002832        0.00000
SCALE2      0.000000  0.104592  0.000000        0.00000
SCALE3      0.000000  0.000000  0.048319        0.00000
ATOM      1  N   LYS A   1       8.009   0.080   3.795  1.00 12.85           N
ATOM      2  CA  LYS A   1       7.501  -0.641   4.336  1.00 11.37           C
ATOM      3  C   LYS A   1       6.186   0.030   3.984  1.00 11.88           C
ATOM      4  O   LYS A   1       6.176   1.208   3.641  1.00 11.02           O
ATOM      5  CB  LYS A   1       7.606  -0.792   5.871  1.00 15.61           C
ATOM      6  CG  LYS A   1       7.543   0.531   6.652  1.00 23.32           C
ATOM      7  CD  LYS A   1       8.894   0.917   7.175  1.00 22.74           C
ATOM      8  CE  LYS A   1       8.851   2.260   7.839  1.00 22.96           C
ATOM      9  NZ  LYS A   1      10.091   2.510   8.611  1.00 18.89           N
ATOM     10  N   LEU A   2       5.063  -0.702   4.139  1.00  8.33           N
ATOM     11  CA  LEU A   2       3.720  -0.164   3.917  1.00  8.69           C
ATOM     12  C   LEU A   2       2.701  -0.803   4.885  1.00  9.37           C
ATOM     13  O   LEU A   2       2.675  -2.017   5.019  1.00  8.16           O
ATOM     14  CB  LEU A   2       3.331  -0.407   2.441  1.00  9.53           C
ATOM     15  CG  LEU A   2       1.966   0.033   1.872  1.00 16.06           C
ATOM     16  CD1 LEU A   2       0.773  -0.824   2.351  1.00 15.86           C
ATOM     17  CD2 LEU A   2       1.747   1.479   1.927  1.00 17.67           C
ATOM     18  N   VAL A   3       1.856   0.019   5.527  1.00  6.29           N
ATOM     19  CA  VAL A   3       0.748  -0.404   6.386  1.00  6.45           C
ATOM     20  C   VAL A   3      -0.501   0.203   5.783  1.00 11.86           C
ATOM     21  O   VAL A   3      -0.569   1.425   5.609  1.00 10.67           O
ATOM     22  CB  VAL A   3       0.865   0.064   7.854  1.00 10.59           C
ATOM     23  CG1 VAL A   3      -0.285  -0.493   8.710  1.00 10.71           C
ATOM     24  CG2 VAL A   3       2.219  -0.288   8.450  1.00 10.64           C
ATOM     25  N   PHE A   4      -1.510  -0.637   5.537  1.00  8.41           N
ATOM     26  CA  PHE A   4      -2.797  -0.223   4.998  1.00  8.51           C
ATOM     27  C   PHE A   4      -3.900  -0.807   5.880  1.00 11.98           C
ATOM     28  O   PHE A   4      -3.871  -2.003   6.216  1.00 10.68           O
ATOM     29  CB  PHE A   4      -2.954  -0.769   3.564  1.00  9.14           C
ATOM     30  CG  PHE A   4      -4.317  -0.526   2.953  1.00  9.69           C
ATOM     31  CD1 PHE A   4      -4.536   0.563   2.127  1.00 10.70           C
ATOM     32  CD2 PHE A   4      -5.362  -1.422   3.159  1.00 12.10           C
ATOM     33  CE1 PHE A   4      -5.787   0.791   1.562  1.00 11.47           C
ATOM     34  CE2 PHE A   4      -6.618  -1.195   2.592  1.00 15.42           C
ATOM     35  CZ  PHE A   4      -6.815  -0.096   1.778  1.00 13.27           C
ATOM     36  N   PHE A   5      -4.902   0.009   6.183  1.00  8.96           N
ATOM     37  CA  PHE A   5      -6.081  -0.452   6.908  1.00  9.67           C
ATOM     38  C   PHE A   5      -7.304   0.208   6.324  1.00 13.65           C
ATOM     39  O   PHE A   5      -7.349   1.428   6.235  1.00 10.13           O
ATOM     40  CB  PHE A   5      -5.986  -0.239   8.445  1.00 11.19           C
ATOM     41  CG  PHE A   5      -7.306  -0.481   9.152  1.00 12.79           C
ATOM     42  CD1 PHE A   5      -7.716  -1.766   9.473  1.00 16.12           C
ATOM     43  CD2 PHE A   5      -8.156   0.580   9.457  1.00 15.48           C
ATOM     44  CE1 PHE A   5      -8.953  -1.987  10.088  1.00 17.28           C
ATOM     45  CE2 PHE A   5      -9.379   0.359  10.098  1.00 18.56           C
ATOM     46  CZ  PHE A   5      -9.771  -0.922  10.402  1.00 15.89           C
ATOM     47  N   ALA A   6      -8.282  -0.608   5.899  1.00 12.21           N
ATOM     48  CA  ALA A   6      -9.570  -0.134   5.389  1.00 15.49           C
ATOM     49  C   ALA A   6     -10.647  -0.909   6.127  1.00 31.54           C
ATOM     50  O   ALA A   6     -10.637  -2.159   6.058  1.00 33.30           O
ATOM     51  CB  ALA A   6      -9.677 [illegible] [illegible] 1.00 [illegible]    C
ATOM     52  OXT ALA A   6     -11.417  -0.275   6.874  1.00 51.39           O
TER      53      ALA A   6
ATOM     54  N   LYS B   1     -10.065  -4.813   6.193  1.00 12.06           N
ATOM     55  CA  LYS B   1      -8.858  -5.207   5.451  1.00 12.29           C
```

we can use the command to grep information we need.

grep “ A ” 3OW9.pdb | grep “ATOM” > monomer.pdb

Then we open monomer.pdb to delete the line having the word “OXT” and save the file.

3) Prepare the topology files for simulation by GROMACS command pdb2max,

source ~ROOT/gromacs4/bin/GMXRC

export GMXLIB=~ROOT/gromacs4/share/gromacs/top/

pdb2gmx -f monomer.pdb -o mono-pace.pdb -p draft.top -ter -ignh and after executing the command, the program needs you to provide three choices:

a. Protein part → "PACE-ASM force field";

b. Solvent part → "cgWater coarse-grained water";

c. Terminal residue type → "0", for the peptide is uncapped.

After having finished this operation, two files are obtained. One is *mono-pace.pdb*, which is used by PACE, and another is *draft.top*, which is used to generate topology file later. The commands are as follows,

```
./genPairPACE count_atom count_residue mono-pace.pdb 1 > mono.patch
```

where count_atom and count_residue is the atom number and residue number in the *mono-pace.pdb*, respectively. Number 1 on the left of ">" represents the peptide is uncapped. Then this command can give the topology file – *pace.top*:

```
python insert_param.py mono.patch draft.top > pace.top
```

According to the manual, for the uncapped peptide, we should modify the force constant at N termini,

```
1    2    8    10    1    -0.0    1.0    1 → 1    2    8    10    1    -0.0    4.0    1
```

and add a CMAP potential at C termini in the *pace.top* file.

```
[ cmap ]
 49    51    53    55    56    1
```

4) Based on obtained coordination file and topology file, we can get a trimer / 18-mers simulation system by GROMACS 5.x command insert-molecules,

```
source ~ROOT/gromacs5/bin/GMXRC
export GMXLIB=~ROOT/gromacs5/share/gromacs/top/
gmx insert-molecules -ci mono-pace.pdb -nmol 3/18 -box 9 9 9 -o trimer/18.gro
```

and can get corresponding topology file by changing the line in the *pace.top*.

```
Protein_chain_A 1 → Protein_chain_A 3/18
```

5) After these files are ready, the following steps are standard procedure of PACE at environment of GROMACS 4.x.

```
source ~ROOT/gromacs4/bin/GMXRC
export GMXLIB=~ROOT/gromacs4/share/gromacs/top/
```

a. Generate box and add CG water into the system.

```
editconf -f trimer.gro/18.gro -o box.gro -c -box 9 9 9
```

```
genbox -cp box.gro -cs cg216water.gro -p pace.top -vdwd 0.235 -o sov.gro
```

b. Add ions to keep system under a salt concentration at 0.15 M

```
grompp -v -f em-posres.mdp -c sov.gro -p pace.top -o sov.tpr
genion -s sov.tpr -o ion.gro -conc 0.15 -neutral -pname NA -nname CL -p pace.top
```

c. Energy minimization with and without position restrain.

```
grompp -v -f em-posres.mdp -c ion.gro -p pace.top -o em-posres.tpr
mpirun mdrun_mpi -s em-posres.tpr -c em-posres.gro

grompp -v -f em.mdp -c em-posres.gro -p pace.top -o em.tpr
mpirun mdrun_mpi -s em.tpr -c em.gro
```

d. Pre-equilibrium at NVT condition

```
grompp -v -f nvt.mdp -c em.gro -p pace.top -o nvt.tpr
mpirun mdrun_mpi -s nvt.tpr -c nvt.gro
```

e. Pre-equilibrium at NPT condition

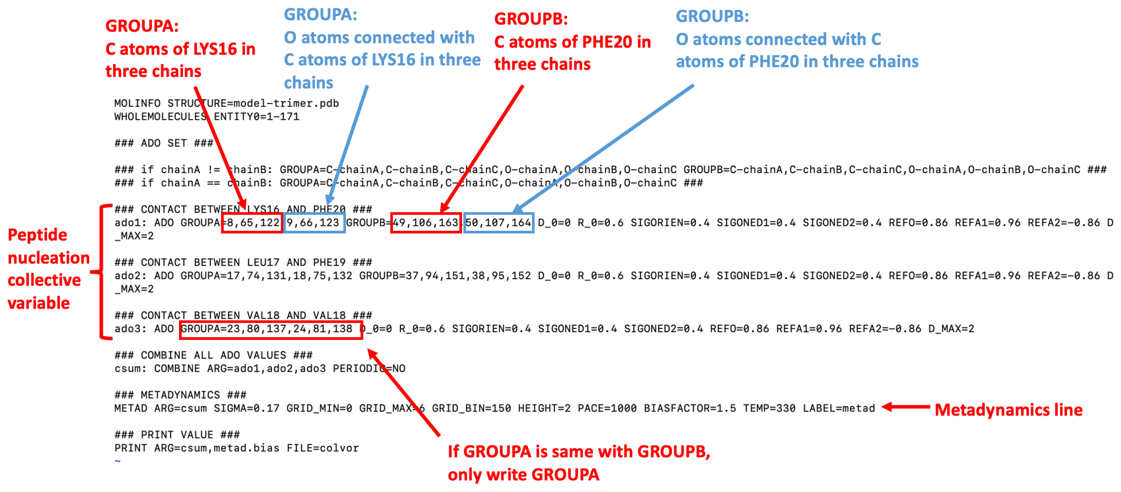
```
grompp -v -f npt.mdp -c nvt.gro -p pace.top -o npt.tpr
mpirun mdrun_mpi -s npt.tpr -c npt.gro
```

f. Production run file generated at NPT condition

```
grompp -v -f full.mdp -c npt.gro -p pace.top -o md.tpr
```

6) Finally, we need to set meta.dat file to do metadynamics simulation.

a. For the trimer system, we set *meta.dat* as follows.

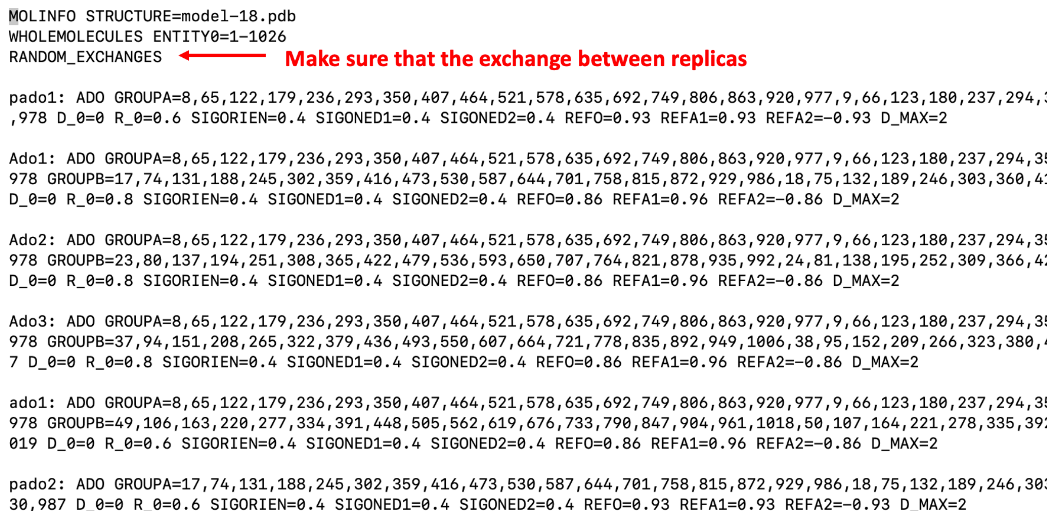

GROUPA:
C atoms of LYS16 in three chains
GROUPA:
O atoms connected with C atoms of LYS16 in three chains
GROUPB:
C atoms of PHE20 in three chains
GROUPB:
O atoms connected with C atoms of PHE20 in three chains
Peptide nucleation collective variable
Metadynamics line
If GROUPA is same with GROUPB, only write GROUPA

```
MOLINFO STRUCTURE=model-trimer.pdb
WHOLEMOLECULES ENTITY0=1-171

### ADO SET ###

### if chainA != chainB: GROUPA=C-chainA,C-chainB,C-chainC,O-chainA,O-chainB,O-chainC GROUPB=C-chainA,C-chainB,C-chainC,O-chainA,O-chainB,O-chainC ###
### if chainA == chainB: GROUPA=C-chainA,C-chainB,C-chainC,O-chainA,O-chainB,O-chainC ###

### CONTACT BETWEEN LYS16 AND PHE20 ###
ado1: ADO GROUPA=8,65,122,9,66,123 GROUPB=49,106,163,50,107,164 D_0=0 R_0=0.6 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D
_MAX=2

### CONTACT BETWEEN LEU17 AND PHE19 ###
ado2: ADO GROUPA=17,74,131,18,75,132 GROUPB=37,94,151,38,95,152 D_0=0 R_0=0.6 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D
_MAX=2

### CONTACT BETWEEN VAL18 AND VAL18 ###
ado3: ADO GROUPA=23,80,137,24,81,138 D_0=0 R_0=0.6 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D_MAX=2

### COMBINE ALL ADO VALUES ###
csum: COMBINE ARG=ado1,ado2,ado3 PERIODIC=NO

### METADYNAMICS ###
METAD ARG=csum SIGMA=0.17 GRID_MIN=0 GRID_MAX=6 GRID_BIN=150 HEIGHT=2 PACE=1000 BIASFACTOR=1.5 TEMP=330 LABEL=metad

### PRINT VALUE ###
PRINT ARG=csum,metad.bias FILE=colvor
~
```

When we having *meta.dat* file, we start our metadynamics simulation.

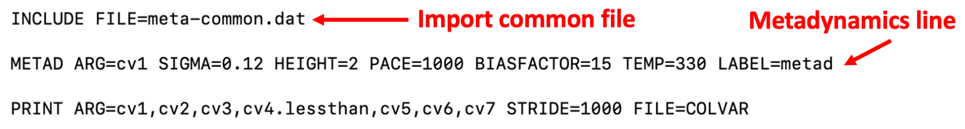
```
mpirun mdrun_mpi -deffnm md -plumed meta.dat -rdd 1.9 -dds 0.9
```

b. For the 18-mer system, we do bias-exchange metadynamics simulation. We first duplicate *md.tpr* into *md0.tpr*, *md1.tpr*, *md2.tpr*, *md3.tpr*, …, *md7.tpr*. Then we set common *meta-common.dat* used to do simulation. This file includes all collective variables used in the simulation without metadynamics line, which is shown as the figure below.



```
MOLINFO STRUCTURE=model-18.pdb
WHOLEMOLECULES ENTITY0=1-1026
RANDOM_EXCHANGES

pado1: ADO GROUPA=8,65,122,179,236,293,350,407,464,521,578,635,692,749,806,863,920,977,9,66,123,180,237,294,
,978 D_0=0 R_0=0.6 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.93 REFA1=0.93 REFA2=-0.93 D_MAX=2

Ado1: ADO GROUPA=8,65,122,179,236,293,350,407,464,521,578,635,692,749,806,863,920,977,9,66,123,180,237,294,3
978 GROUPB=17,74,131,188,245,302,359,416,473,530,587,644,701,758,815,872,929,986,18,75,132,189,246,303,360,4
D_0=0 R_0=0.8 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D_MAX=2

Ado2: ADO GROUPA=8,65,122,179,236,293,350,407,464,521,578,635,692,749,806,863,920,977,9,66,123,180,237,294,3
978 GROUPB=23,80,137,194,251,308,365,422,479,536,593,650,707,764,821,878,935,992,24,81,138,195,252,309,366,4
D_0=0 R_0=0.8 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D_MAX=2

Ado3: ADO GROUPA=8,65,122,179,236,293,350,407,464,521,578,635,692,749,806,863,920,977,9,66,123,180,237,294,3
978 GROUPB=37,94,151,208,265,322,379,436,493,550,607,664,721,778,835,892,949,1006,38,95,152,209,266,323,380,
7 D_0=0 R_0=0.8 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D_MAX=2

ado1: ADO GROUPA=8,65,122,179,236,293,350,407,464,521,578,635,692,749,806,863,920,977,9,66,123,180,237,294,3
978 GROUPB=49,106,163,220,277,334,391,448,505,562,619,676,733,790,847,904,961,1018,50,107,164,221,278,335,39
019 D_0=0 R_0=0.6 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.86 REFA1=0.96 REFA2=-0.86 D_MAX=2

pado2: ADO GROUPA=17,74,131,188,245,302,359,416,473,530,587,644,701,758,815,872,929,986,18,75,132,189,246,30
30,987 D_0=0 R_0=0.6 SIGORIEN=0.4 SIGONED1=0.4 SIGONED2=0.4 REFO=0.93 REFA1=0.93 REFA2=-0.93 D_MAX=2
```

Include this *meta-common.dat* file into different files *meta.0.dat*, *meta.1.dat*, *meta.2.dat, …*, *meta.7.dat*. And the metadynamics line is written in theses file, which is shown as this figure.



```
INCLUDE FILE=meta-common.dat

METAD ARG=cv1 SIGMA=0.12 HEIGHT=2 PACE=1000 BIASFACTOR=15 TEMP=330 LABEL=metad

PRINT ARG=cv1,cv2,cv3,cv4.lessthan,cv5,cv6,cv7 STRIDE=1000 FILE=COLVAR
```

We can do bias-exchange metadynamics by this command.

mpirun mdrun_mpi -deffnm md -rdd 2.0 -dds 0.9 -maxh 24.5 -plumed meta -multi 8 -replex 2000

Free energy construction of peptide nucleation of Aβ16-21

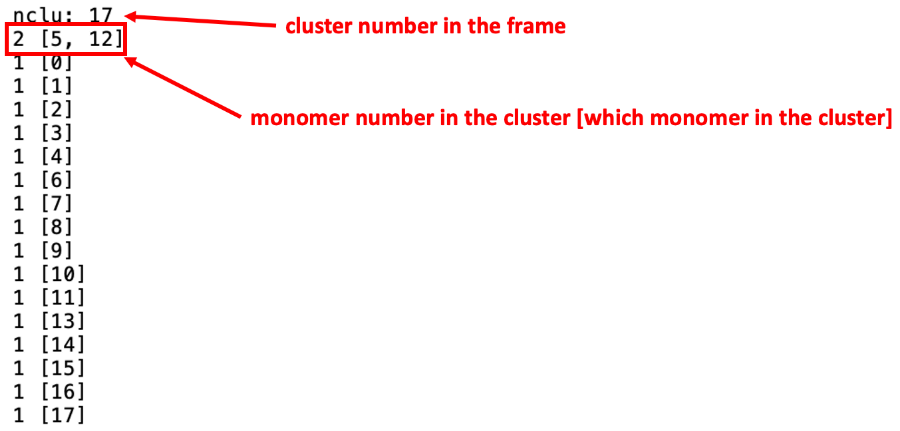
In this section, we will show how to use scripts in *Free Energy Construction/* directory to obtain free energy respect to oligomer size, β-strands number n_β and maximum burial depth of misregistered β-strands m. Again, we assume that the current directory contains all the files from *~WORK/Peptide_Nucleation/Free Energy Construction/*.

1) We first do cluster from the simulation trajectory:

   python *Scripts*/cal-cluster.py file_pdb.pdb file_trajectory.xtc > clu_result.xvg

   We should have two files generated, *file_pdb.pdb* of peptides in the simulation system and *file_trajectory.xtc* from the simulation. Here we provide example files in folder *~WORK/Peptide_Nucleation/Free Energy Construction/Example/.*

   The output file *clu_result.xvg* is shown as the figure below.



```
nclu: 17
2 [5, 12]
1 [0]
1 [1]
1 [2]
1 [3]
1 [4]
1 [6]
1 [7]
1 [8]
1 [9]
1 [10]
1 [11]
1 [13]
1 [14]
1 [15]
1 [16]
1 [17]
```

cluster number in the frame
monomer number in the cluster [which monomer in the cluster]

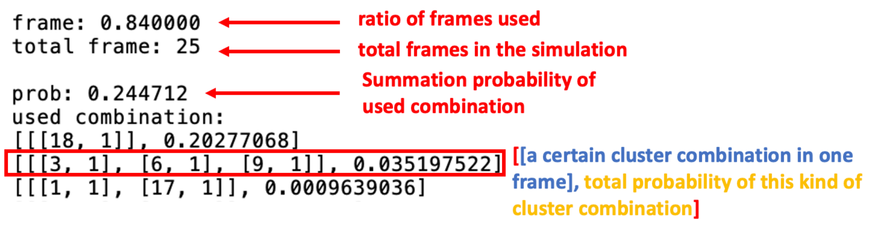
   And from *clu_result.xvg* file, we summarize the frames of having same number of oligomers in a cluster into same file *frame.xvg*, and for each of the frames in the file we record corresponding monomer indexes in the cluster into file *clu.xvg*. These two files will be used in step 3.

2) Next, we calculate the free energy respect to oligomer size $F(n)$ by

   python *Scripts*/re-boot_strapping-frame.py num_boot_strapping diff

   where num_boot_strapping is the number of doing boot strapping sampling, here we use 5000 as an example, and diff represents the removal of numbers of partitioning states, usually between 0.1~2, the value of diff is smaller, the removal is more. And the *weight.dat* from simulation if you do metadynamics should be prepared as an input file. According to this script, we obtain three files.

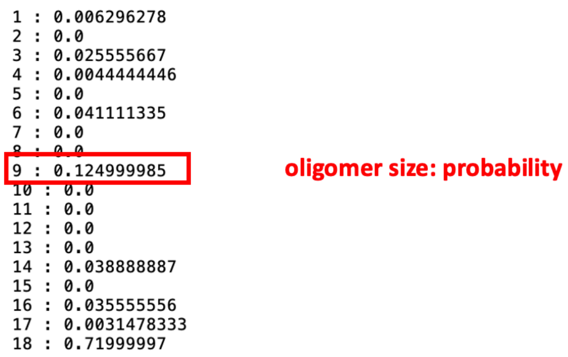
   a. File *prob.xvg* gives the ratio of frame and which combination of different clusters used when calculate the free energy;



```
frame: 0.840000
total frame: 25

prob: 0.244712
used combination:
[[[18, 1]], 0.20277068]
[[[3, 1], [6, 1], [9, 1]], 0.035197522]
[[[1, 1], [17, 1]], 0.0009639036]
```

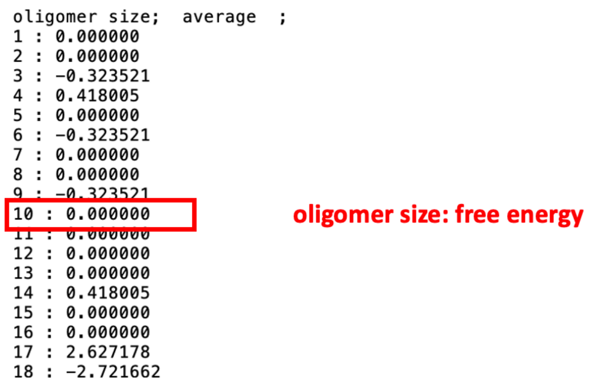
ratio of frames used
total frames in the simulation
Summation probability of used combination
[[a certain cluster combination in one frame], total probability of this kind of cluster combination]

b. File *oligomer-distribution.xvg* gives the probability of different size oligomers



```
1 : 0.006296278
2 : 0.0
3 : 0.025555667
4 : 0.0044444446
5 : 0.0
6 : 0.041111335
7 : 0.0
8 : 0.0
9 : 0.124999985
10 : 0.0
11 : 0.0
12 : 0.0
13 : 0.0
14 : 0.038888887
15 : 0.0
16 : 0.035555556
17 : 0.0031478333
18 : 0.71999997
```

c. File *relative-free-ene-ave.xvg* gives the free energy of different oligomer size respect to monomer

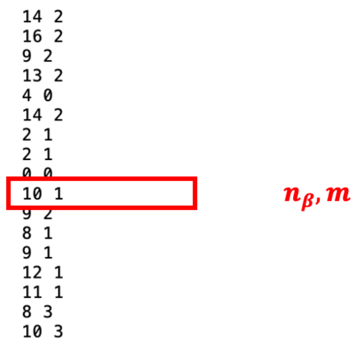


```
oligomer size;  average  ;
1 : 0.000000
2 : 0.000000
3 : -0.323521
4 : 0.418005
5 : 0.000000
6 : -0.323521
7 : 0.000000
8 : 0.000000
9 : -0.323521
10 : 0.000000
11 : 0.000000
12 : 0.000000
13 : 0.000000
14 : 0.418005
15 : 0.000000
16 : 0.000000
17 : 2.627178
18 : -2.721662
```

3) We then calculate the probability of combination $[n_\beta, m]$ in different oligomer size n, $P_{eq,n}(n_\beta, m)$:

python *Scripts*/cal_buried_rc.py file_pdb.pdb file_trajectory.xtc n n_pep

where n is the number of monomers in the cluster, and n_pep is the number of monomers in the simulation system. Files *file_pdb.pdb* and *file_trajectory.xtc* are same with step 1. And the output file *beta_depth.xvg* files for different oligomer size n are as below.



```
14 2
16 2
9 2
13 2
4 0
14 2
2 1
2 1
0 0
10 1
9 2
8 1
9 1
12 1
11 1
8 3
10 3
```

According to these *beta_depth.xvg* files, we can get the distribution of $P_{eq,n}(n_\beta, m)$ for different oligomer size n by

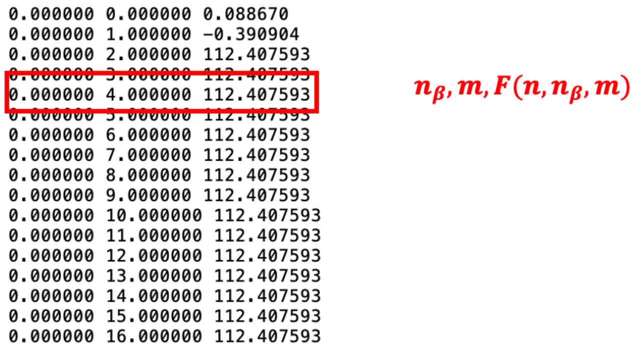
python *Scripts*/distribution.py n > prob-beta-depth.xvg

where n is the oligomer size. In addition to *beta_depth.xvg*, the input files are weight.dat, *frame.xvg* and *clu.xvg* obtained in step 1.

4) Finally, based on $F(n)$ from step 2 and $P_{eq,n}(n_\beta, m)$ from step 3, we calculate the free energy as $F(n, n_\beta, m) = -kTlog(P_{n,eq}(n_\beta, m)) + F(n)$

python *Scripts*/get_f.py relative-free-ene-ave.xvg prob-beta-depth.xvg n

where file *relative-free-ene-ave.xvg* obtained from step 2, *prob-beta-depth.xvg* obtained from step 3 and n is the oligomer size. The output files are for different oligomer size, and one of them is as below

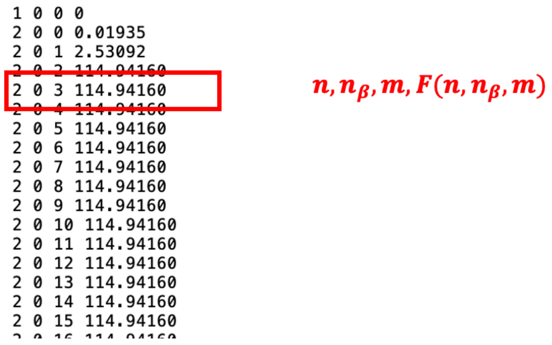


```
0.000000 0.000000 0.088670
0.000000 1.000000 -0.390904
0.000000 2.000000 112.407593
0.000000 3.000000 112.407593
0.000000 4.000000 112.407593
0.000000 5.000000 112.407593
0.000000 6.000000 112.407593
0.000000 7.000000 112.407593
0.000000 8.000000 112.407593
0.000000 9.000000 112.407593
0.000000 10.000000 112.407593
0.000000 11.000000 112.407593
0.000000 12.000000 112.407593
0.000000 13.000000 112.407593
0.000000 14.000000 112.407593
0.000000 15.000000 112.407593
0.000000 16.000000 112.407593
```

Kinetic transition network building of peptide nucleation of Aβ16-21

In this section, we will show how to use scripts in *Kinetic Transition Network/* directory to build kinetic transition network based on the free energy $F(n, n_\beta, m)$ from last step. And according to the network, we can further do Monte Carlo simulations to get the evolution of peptide nucleation and the mean first passage time of this process. Again, we assume that the current directory contains all the files from *~WORK/Peptide_Nucleation/Kinetic Transition Network/*.

1) We first combine the free energy files of $F(n, n_\beta, m)$ from last step in different oligomer size into one file, which is shown as below



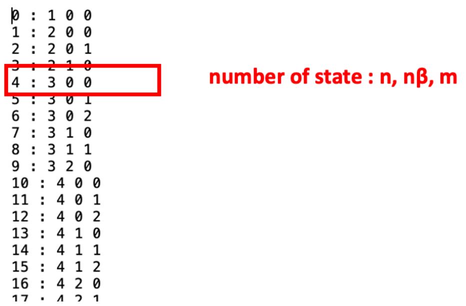
```
1 0 0 0
2 0 0 0.01935
2 0 1 2.53092
2 0 2 114.94160
2 0 3 114.94160
2 0 4 114.94160
2 0 5 114.94160
2 0 6 114.94160
2 0 7 114.94160
2 0 8 114.94160
2 0 9 114.94160
2 0 10 114.94160
2 0 11 114.94160
2 0 12 114.94160
2 0 13 114.94160
2 0 14 114.94160
2 0 15 114.94160
2 0 16 114.94160
```

2) We then use the combined file in step 1 as input file, and the concentration, which unit is μM, of this condition as input parameter to build kinetic network, catch the states and their free energies in the network, and calculate the rate constant of each transition between states.

   python *Scripts*/map_network_rate.py combined_free_energy.xvg concentration

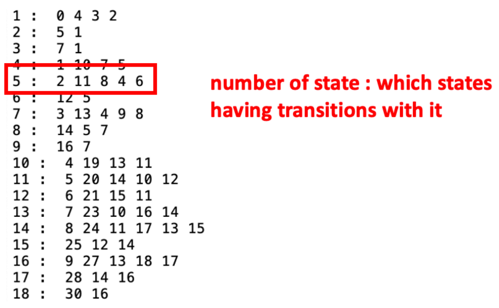
   There are four output files.

   a. File *crd.xvg* provides the coordination of each state;



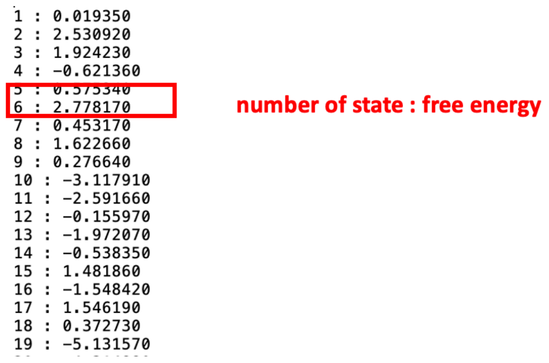
```
0 : 1 0 0
1 : 2 0 0
2 : 2 0 1
3 : 2 1 0
4 : 3 0 0
5 : 3 0 1
6 : 3 0 2
7 : 3 1 0
8 : 3 1 1
9 : 3 2 0
10 : 4 0 0
11 : 4 0 1
12 : 4 0 2
13 : 4 1 0
14 : 4 1 1
15 : 4 1 2
16 : 4 2 0
17 : 4 2 1
```

   b. File *network.xvg* provides which states have transitions;



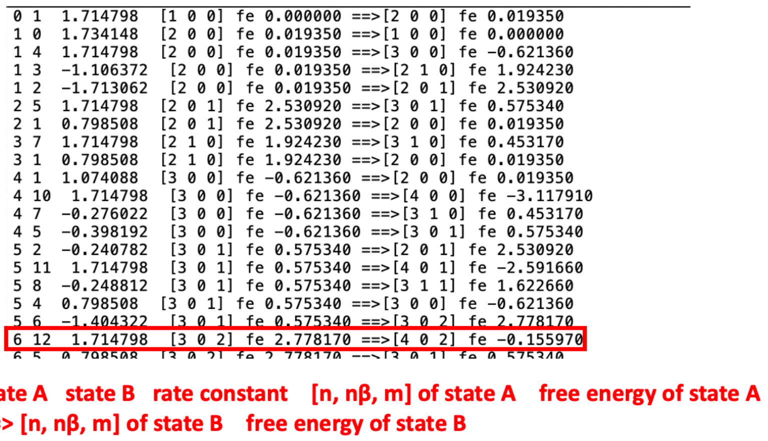
```
1 :  0 4 3 2
2 :  5 1
3 :  7 1
4 :  1 10 7 5
5 :  2 11 8 4 6
6 :  12 5
7 :  3 13 4 9 8
8 :  14 5 7
9 :  16 7
10 :  4 19 13 11
11 :  5 20 14 10 12
12 :  6 21 15 11
13 :  7 23 10 16 14
14 :  8 24 11 17 13 15
15 :  25 12 14
16 :  9 27 13 18 17
17 :  28 14 16
18 :  30 16
```

c. File *fe.xvg* provides the free energy of each state;



```
1 : 0.019350
2 : 2.530920
3 : 1.924230
4 : -0.621360
5 : 0.575340
6 : 2.778170
7 : 0.453170
8 : 1.622660
9 : 0.276640
10 : -3.117910
11 : -2.591660
12 : -0.155970
13 : -1.972070
14 : -0.538350
15 : 1.481860
16 : -1.548420
17 : 1.546190
18 : 0.372730
19 : -5.131570
```

d. File *k_const.xvg* provides the summaries of the first three files and the rate constant of each transition;



```
0 1  1.714798  [1 0 0] fe 0.000000 ==>[2 0 0] fe 0.019350
1 0  1.734148  [2 0 0] fe 0.019350 ==>[1 0 0] fe 0.000000
1 4  1.714798  [2 0 0] fe 0.019350 ==>[3 0 0] fe -0.621360
1 3  -1.106372  [2 0 0] fe 0.019350 ==>[2 1 0] fe 1.924230
1 2  -1.713062  [2 0 0] fe 0.019350 ==>[2 0 1] fe 2.530920
2 5  1.714798  [2 0 1] fe 2.530920 ==>[3 0 1] fe 0.575340
2 1  0.798508  [2 0 1] fe 2.530920 ==>[2 0 0] fe 0.019350
3 7  1.714798  [2 1 0] fe 1.924230 ==>[3 1 0] fe 0.453170
3 1  0.798508  [2 1 0] fe 1.924230 ==>[2 0 0] fe 0.019350
4 1  1.074088  [3 0 0] fe -0.621360 ==>[2 0 0] fe 0.019350
4 10  1.714798  [3 0 0] fe -0.621360 ==>[4 0 0] fe -3.117910
4 7  -0.276022  [3 0 0] fe -0.621360 ==>[3 1 0] fe 0.453170
4 5  -0.398192  [3 0 0] fe -0.621360 ==>[3 0 1] fe 0.575340
5 2  -0.240782  [3 0 1] fe 0.575340 ==>[2 0 1] fe 2.530920
5 11  1.714798  [3 0 1] fe 0.575340 ==>[4 0 1] fe -2.591660
5 8  -0.248812  [3 0 1] fe 0.575340 ==>[3 1 1] fe 1.622660
5 4  0.798508  [3 0 1] fe 0.575340 ==>[3 0 0] fe -0.621360
5 6  -1.404322  [3 0 1] fe 0.575340 ==>[3 0 2] fe 2.778170
6 12  1.714798  [3 0 2] fe 2.778170 ==>[4 0 2] fe -0.155970
6 5  0.798508  [3 0 2] fe 2.778170 ==>[3 0 1] fe 0.575340
```

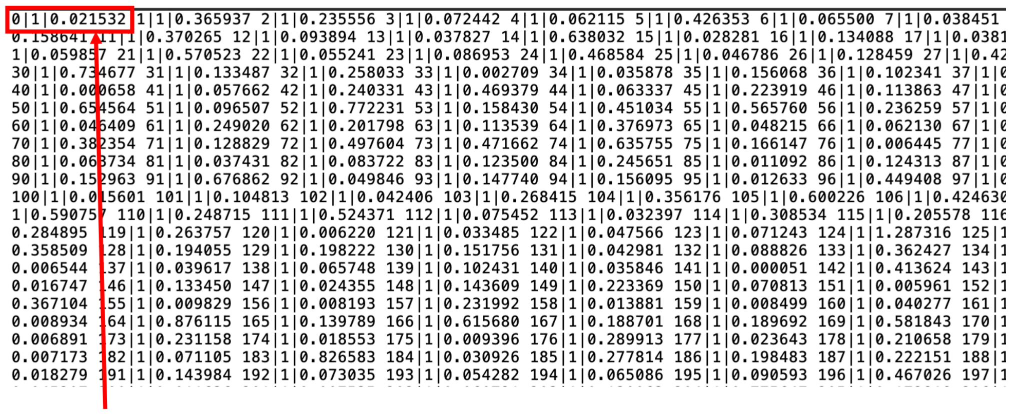
state A   state B   rate constant   [n, nβ, m] of state A   free energy of state A
==> [n, nβ, m] of state B   free energy of state B

3) We can use the information in file *k_const.xvg* to do Monte Carlo simulations.

python *Scripts*/KMC.py k_const.xvg start_point end_point num_traj > mc.log

where start_point and end_point is the begin and end state for the simulation, respectively, and num_traj is how many trajectories running in the simulation.

For the output file mc.log, units with number of MC trajectories are shown in each line. In each unit, there are three parts shown as figure below.



```
0|1|0.021532 1|1|0.365937 2|1|0.235556 3|1|0.072442 4|1|0.062115 5|1|0.426353 6|1|0.065500 7|1|0.038451
0.158641 11|1|0.370265 12|1|0.093894 13|1|0.037827 14|1|0.638032 15|1|0.028281 16|1|0.134088 17|1|0.0381
1|0.059857 21|1|0.570523 22|1|0.055241 23|1|0.086953 24|1|0.468584 25|1|0.046786 26|1|0.128459 27|1|0.42
30|1|0.734677 31|1|0.133487 32|1|0.258033 33|1|0.002709 34|1|0.035878 35|1|0.156068 36|1|0.102341 37|1|0
40|1|0.000658 41|1|0.057662 42|1|0.240331 43|1|0.469379 44|1|0.063337 45|1|0.223919 46|1|0.113863 47|1|0
50|1|0.654564 51|1|0.096507 52|1|0.772231 53|1|0.158430 54|1|0.451034 55|1|0.565760 56|1|0.236259 57|1|0
60|1|0.046409 61|1|0.249020 62|1|0.201798 63|1|0.113539 64|1|0.376973 65|1|0.048215 66|1|0.062130 67|1|0
70|1|0.382354 71|1|0.128829 72|1|0.497604 73|1|0.471662 74|1|0.635755 75|1|0.166147 76|1|0.006445 77|1|0
80|1|0.068734 81|1|0.037431 82|1|0.083722 83|1|0.123500 84|1|0.245651 85|1|0.011092 86|1|0.124313 87|1|0
90|1|0.152963 91|1|0.676862 92|1|0.049846 93|1|0.147740 94|1|0.156095 95|1|0.012633 96|1|0.449408 97|1|0
100|1|0.015601 101|1|0.104813 102|1|0.042406 103|1|0.268415 104|1|0.356176 105|1|0.600226 106|1|0.42463
1|0.590757 110|1|0.248715 111|1|0.524371 112|1|0.075452 113|1|0.032397 114|1|0.308534 115|1|0.205578 116
0.284895 119|1|0.263757 120|1|0.006220 121|1|0.033485 122|1|0.047566 123|1|0.071243 124|1|1.287316 125|1
0.358509 128|1|0.194055 129|1|0.198222 130|1|0.151756 131|1|0.042981 132|1|0.088826 133|1|0.362427 134|1
0.006544 137|1|0.039617 138|1|0.065748 139|1|0.102431 140|1|0.035846 141|1|0.000051 142|1|0.413624 143|1
0.016747 146|1|0.133450 147|1|0.024355 148|1|0.143609 149|1|0.223369 150|1|0.070813 151|1|0.005961 152|1
0.367104 155|1|0.009829 156|1|0.008193 157|1|0.231992 158|1|0.013881 159|1|0.008499 160|1|0.040277 161|1
0.008934 164|1|0.876115 165|1|0.139789 166|1|0.615680 167|1|0.188701 168|1|0.189692 169|1|0.581843 170|1
0.006891 173|1|0.231158 174|1|0.018553 175|1|0.009396 176|1|0.289913 177|1|0.023643 178|1|0.210658 179|1
0.007173 182|1|0.071105 183|1|0.826583 184|1|0.030926 185|1|0.277814 186|1|0.198483 187|1|0.222151 188|1
0.018279 191|1|0.143984 192|1|0.073035 193|1|0.054282 194|1|0.065086 195|1|0.090593 196|1|0.467026 197|1
```

a|b|c, where a is which MC trajectory runs, b is which state at the moment in the MC and c is the time MC runs at the moment.

When one trajectory has reached the end state, a sentence "* trjs left" occurs as `0 trjs left` , where * is number of num_traj minus trajectories have finished. And at the end of *mc.log*, the mfpt is shown as `mfpt 44.593281667042724` . Of note, the mfpt in mc.log should be divided by 10 to get the actual time, and the unit of time is μs.