Running BE-META Simulations on SLURM HPC with GROMACS 2018 (Written by Aidan Fike, updated for GROMACS 2018 by Jovan Damjanovic)

Before reading: This process is meant to be completed after you already have an equilibrated solvated structure, represented by an equilibrated gro file and corresponding topology file.

If you are not familiar with BE-META simulations as a concept, it will be good to first read literature about the theory. Reading “Escaping free-energy minima” by Alessandro Laio followed by “A Bias-Exchange Approach to Protein Folding” by Stefano Piana and Alessandro Laio should give you a solid foundation (they’re somewhat complicated, but obviously feel free to ask for help). Also, reading some of our lab’s papers will help. One of the most relevant I’ve found is “Insights into How Cyclic Peptides Switch Conformations” written by Sean M. McHugh.

Part 1: Getting Access to Plumed and Needed Modules

Plumed is an implementation of Bias-Exchange Metadynamics which is compatible with gromacs. We will use this package to simulate cyclic peptides. To get plumed on the cluster, the group has been currently using Jovan’s runtime-compiled build at

/cluster/tufts/ysl8/jovan/gromacs_gpu/plumed/lib/libplumedKernel.so

To use this implementation, set your PLUMED_KERNEL environment variable in your job submission script using

```
export
PLUMED_KERNEL=/cluster/tufts/ysl8/jovan/gromacs_gpu/plumed/lib/libplumedKernel.s
o
```

Make sure to also source the GROMACS build you intend to use (/cluster/tufts/ysl8/jovan/gromacs_m4/gromacs/bin/GMXRC is the most general one, which will work with all of our lab’s nodes) and load all required modules (in the case of this build, gcc/4.9.2, cuda/8.0.44, openmpi/2.1.2, gsl)

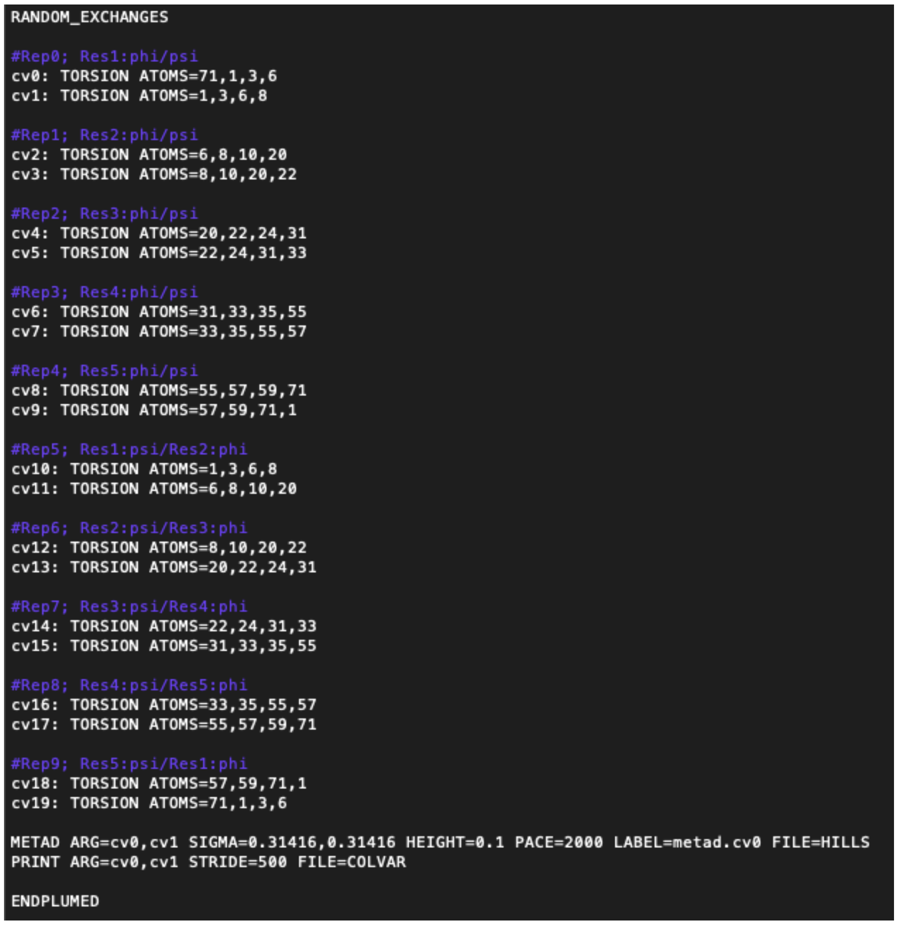
Additionally enter “echo $GMXLIB” to make sure that you are using your desired (and/or local) GMX library. Otherwise, use a command such as

export GMXLIB=/cluster/tufts/ysl8/jovan/gromacs_m4/gromacs/share/gromacs/top/ for a library with files pre-modified for use with our lab’s protocols.

Make sure you do not have other versions of openmpi or gromacs also loaded onto your path before running your simulation. Otherwise the cluster will likely get confused and you risk unintended behavior. Use “module unload ___” to get unload specific modules or “module purge” to get rid of all of your currently-loaded modules.

Part 2: Setting up a Be-Meta simulation

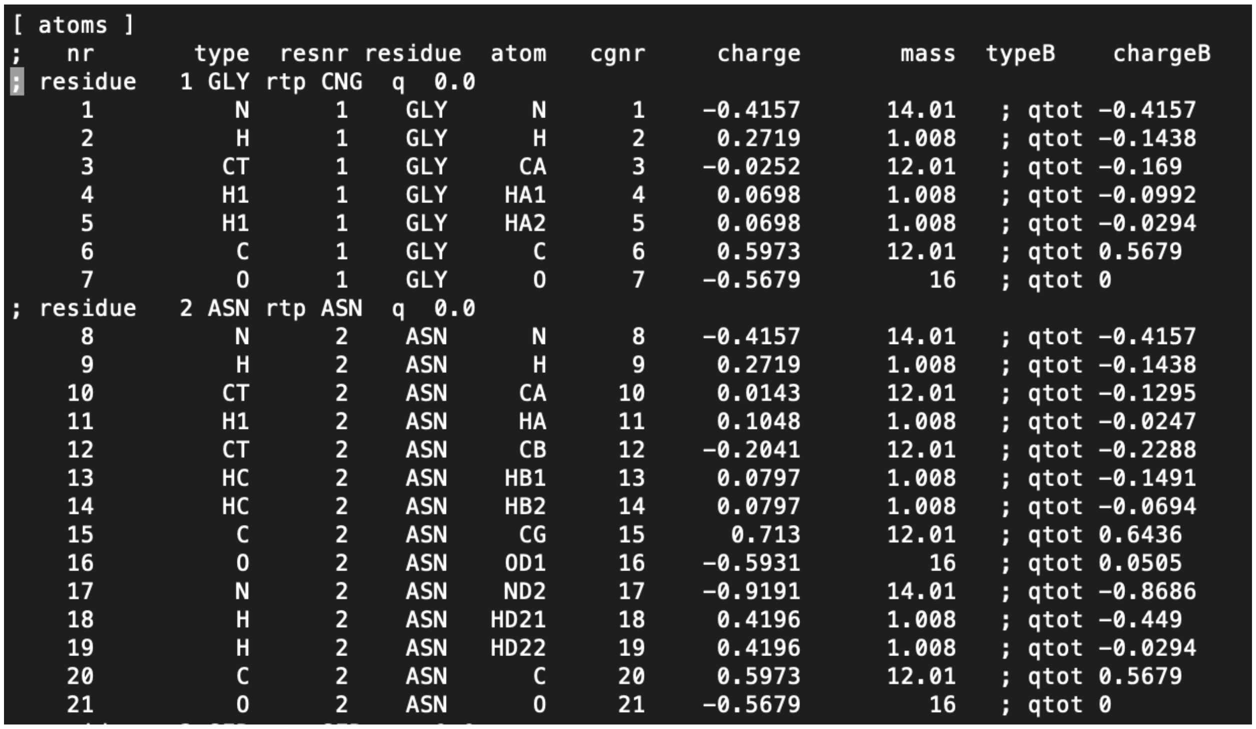
1. First, collect your now-equilibrated gro and topology files that represent your solvated cyclic peptide
2. Next, create your mdp file. These parameters should be nearly identical to the NPT file used for your equilibration. The only major change is that instead of constraining all bonds, only constrain h-bonds (bonds which contain a hydrogen, not the classic “hydrogen bond” you learn about which stabilizes ice). Additionally, you will change the time of the simulation to be 100ns with a 2fs timestep. For the sake of later analysis, it is necessary to output energy, xtc, and log file information every picosecond. An .mdp file can be found in thisrepository and is titled prod_npt_bemeta.mdp.
3. In addition to the classic gro, top, mdp file combination that gromacs utilizes, plumed requires bemeta.dat files. These .dat files will contain information about the CVs (collective variables) that will be biased in each replica. Currently, our lab has found that one of the main obstacles to exploring the free-energy landscape of a cyclic peptide is the ring strain introduced by cyclization. This makes it difficult for a given peptide to switch between conformations. To overcome this, the phi/psi angles of the backbone are biased. Specifically, there will be replicas biasing for the phi/psi combination of each residue as well as replicas biasing the psi angle of each residue paired with the phi angle of the following residue. So, for something like cGNSRV, you will have 10 biased collective-variable pairs - 5 for phi/psi combinations and another 5 for psi/phi+1 combinations. To give this information to plumed, we use a bemeta.dat file. Here is an example of this:



```
RANDOM_EXCHANGES

#Rep0; Res1:phi/psi
cv0: TORSION ATOMS=71,1,3,6
cv1: TORSION ATOMS=1,3,6,8

#Rep1; Res2:phi/psi
cv2: TORSION ATOMS=6,8,10,20
cv3: TORSION ATOMS=8,10,20,22

#Rep2; Res3:phi/psi
cv4: TORSION ATOMS=20,22,24,31
cv5: TORSION ATOMS=22,24,31,33

#Rep3; Res4:phi/psi
cv6: TORSION ATOMS=31,33,35,55
cv7: TORSION ATOMS=33,35,55,57

#Rep4; Res5:phi/psi
cv8: TORSION ATOMS=55,57,59,71
cv9: TORSION ATOMS=57,59,71,1

#Rep5; Res1:psi/Res2:phi
cv10: TORSION ATOMS=1,3,6,8
cv11: TORSION ATOMS=6,8,10,20

#Rep6; Res2:psi/Res3:phi
cv12: TORSION ATOMS=8,10,20,22
cv13: TORSION ATOMS=20,22,24,31

#Rep7; Res3:psi/Res4:phi
cv14: TORSION ATOMS=22,24,31,33
cv15: TORSION ATOMS=31,33,35,55

#Rep8; Res4:psi/Res5:phi
cv16: TORSION ATOMS=33,35,55,57
cv17: TORSION ATOMS=55,57,59,71

#Rep9; Res5:psi/Res1:phi
cv18: TORSION ATOMS=57,59,71,1
cv19: TORSION ATOMS=71,1,3,6

METAD ARG=cv0,cv1 SIGMA=0.31416,0.31416 HEIGHT=0.1 PACE=2000 LABEL=metad.cv0 FILE=HILLS
PRINT ARG=cv0,cv1 STRIDE=500 FILE=COLVAR

ENDPLUMED
```

The first line, RANDOM_EXCHANGES, tells plumed to randomly swap trajectories between replicas. The timing of when these swaps happen is defined by the mdrun call, as to be seen later. The RANDOM_EXCHANGES line just tells plumed that when plumed does stop to exchange trajectories between replicas, it is done so randomly.

The next chunk of lines defines the collective variables used in the simulation. Cv2 is, for example, the phi angle of asparagine in GNSRV. This phi angle is defined by the collection of the 4 atomic indices that correspond to the relevant backbone atoms. We see that for the phi angle of asparagine, we are using indices {6,8,10,20}. Looking at our topology file below, we see that these indices correspond to {C N CA C}, the backbone atoms whose dihedral angle is known as “phi”. Pretty cool!



```
[ atoms ]
;   nr       type  resnr residue  atom   cgnr     charge       mass  typeB    chargeB
; residue   1 GLY rtp CNG  q  0.0
     1          N      1    GLY      N      1    -0.4157      14.01   ; qtot -0.4157
     2          H      1    GLY      H      2     0.2719      1.008   ; qtot -0.1438
     3         CT      1    GLY     CA      3    -0.0252      12.01   ; qtot -0.169
     4         H1      1    GLY    HA1      4     0.0698      1.008   ; qtot -0.0992
     5         H1      1    GLY    HA2      5     0.0698      1.008   ; qtot -0.0294
     6          C      1    GLY      C      6     0.5973      12.01   ; qtot 0.5679
     7          O      1    GLY      O      7    -0.5679         16   ; qtot 0
; residue   2 ASN rtp ASN  q  0.0
     8          N      2    ASN      N      8    -0.4157      14.01   ; qtot -0.4157
     9          H      2    ASN      H      9     0.2719      1.008   ; qtot -0.1438
    10         CT      2    ASN     CA     10     0.0143      12.01   ; qtot -0.1295
    11         H1      2    ASN     HA     11     0.1048      1.008   ; qtot -0.0247
    12         CT      2    ASN     CB     12    -0.2041      12.01   ; qtot -0.2288
    13         HC      2    ASN    HB1     13     0.0797      1.008   ; qtot -0.1491
    14         HC      2    ASN    HB2     14     0.0797      1.008   ; qtot -0.0694
    15          C      2    ASN     CG     15      0.713      12.01   ; qtot 0.6436
    16          O      2    ASN    OD1     16    -0.5931         16   ; qtot 0.0505
    17          N      2    ASN    ND2     17    -0.9191      14.01   ; qtot -0.8686
    18          H      2    ASN   HD21     18     0.4196      1.008   ; qtot -0.449
    19          H      2    ASN   HD22     19     0.4196      1.008   ; qtot -0.0294
    20          C      2    ASN      C     20     0.5973      12.01   ; qtot 0.5679
    21          O      2    ASN      O     21    -0.5679         16   ; qtot 0
```

At the bottom of the bemeta.dat file you will see the METAD and PRINT lines. This file, written using the scripts in this repository, will be sent to a set of replicas in your Be-Meta simulation. The first 10 (or more generally, 2*N*) of these replicas will become the "biased replicas". In each biased replica, a different pair of collective variables (phi/psi or psi/phi+1) are being biased. In the METAD line of the bemeta.dat file, the "ARG=" indicates which two variables are being biased for each replica. The above example shows cv0,cv1 being biased - an actual production file will have multiple sets of CVs in curly brackets, e.g. {cv0,cv1}, {cv2,cv3},... The other parameters of METAD are the same across these biased replicas. SIGMA is the 2D width of the inserted Gaussian hills, and HEIGHT is the hill height. PACE is how often these HILLS are inserted, given in number of timesteps. FILE is the file where information about the inserted HILLS is deposited. Because there are 10 biased replicas in this case, we therefore expect to see 10 different HILLS files being outputted. The PRINT command seen below METAD is simpler - it simply tells plumed to output the values of the indicated collective variables every STRIDE number of steps to FILE.

I mentioned before that there were 15 total bemeta.dat files, but only 10 of them were biased. The other 5 are known as neutral replicas. In these, there are no gaussian hills being inserted. Therefore, the canonical force field remains intact. To tell PLUMED that no Gaussian hills are being inserted, the hill height is zeroed out.

[Edit by Francini: a .tpr file also needs to be created for each replica before running a Be-Meta simulation. You can grompp the .gro file from the last equilibration step (here would be NPT unrestricted) to create one .tpr file and then copy and paste that .tpr file many times as needed for each replica]

Part 3: Running a Be-Meta simulation

Once you have assembled your bemeta.dat files, along with your topology, gro, and mdp files, you are ready to let the computer do the hard work are simulating your system for a couple of days. The actual lines to run your BeMeta simulation are unexpectedly simple:

```
gmx_mpi grompp -v -f prod.mdp -p GNSRV_rsff2_tip3p.top -c equil.gro -o start.tpr &> gmp.log
mpirun gmx_mpi mdrun -v -plumed bemeta -multi 15 -replex 2500 -s start -deffnm prod
```

The grompp command should hopefully be fairly familiar by now. The mpirun gmx_mpi mdrun command is more interesting. This command tells the cluster that you are using multiple, parallel mpi threads using gromacs with plumed. Your -multi 15 argument tells plumed that you want 15 replicas. -replex 2500 tells how many timesteps should be between each replica-exchange process. In this case, since there are 2fs timesteps, we see there are exchanges every 5ps.

If you do not know how to submit jobs on the cluster, check out the example job submission scripts in the parent directory of this repository

Part 4: Extending a Be-Meta simulation

After running your first 100ns simulation on both your s1 and s2 initial conformations, it is important to check out its Normalized Integrated Product (NIP). To understand this, check out the “dPCA and Cluster Analysis” tutorial. This is an important tutorial to do with analyzing the results Be-Meta simulations. To calculate NIP, however, you only need to do up until the section where the phipsi/covar directory is created and the eigenvec.trr file is obtained. As a quick summary, this file contains the eigenvectors from dihedral principal component analysis. In more human terms (although still pretty mathy) this means that the file contains a lower dimensional representation of the protein’s phi/psi angles. So, rather than representing your cyclic backbone with 10 numbers representing the angles of your 10 dihedrals, you are able to instead get away with only using, for example, 3 numbers without losing much information.

Once you have this eigenvec.trr file, you are able to project the phi/psi dihedrals of your protein onto a 3-dimensional space. This makes the trajectories of your s1 and s2 trajectories easily comparable. To see whether your s1 and s2 trajectories have converged onto a given structural ensemble, you will want to see how similar these low-dimensional space representations are. This is what NIP is meant to indicate, where NIP=1 is perfect convergence and NIP=0 is no convergence.

To calculate NIP, first make a 50x50x50 grid of the 3-dimensional trajectory space for each of your s1 and s2 trajectories. Then, calculate the density of each of these cells: the fraction of frames it contains divided by the volume of the cell. Using these densities, you can then calculate NIP as

$$\mathrm{NIP} = \frac{2\sum_i \rho_i \rho_{i,\mathrm{ref}}}{\sum_i \rho_i^2 + \sum_i \rho_{i,\mathrm{ref}}^2}$$

The sigmas indicate that you sum together the various calculations for all of the cells in your grid. The “reference” density, ρ_{ref}, is simply the density of the other simulation’s corresponding cell during the last 25-50ns. So, for s1, you are comparing to the end of s2’s simulation as the “true” distribution and for s2, you are comparing to s1 as the “true” distribution. Whether you choose 25 or 50 nanoseconds depends on the length of your entire simulation. If 200ns or more it is often better to use 50ns, whereas for 100ns it is better to use 25ns. You want to calculate NIP in simulation “chunks”, as the calculated structural ensemble is still being discovered at earlier times. It is nice to compare, for example, the first 25ns of the s1 simulation to the last 25ns of the s2 simulation to see how unconverged s1 is at during that time frame. Then, you will also want to calculate the NIP between the last 25ns of each simulation to see how close they are after a greater number of hills are deposited.

If last 50ns of your simulation has a low NIP (where “low” is usually less than 0.9). If this is the case, your system is gradually converging on a solution, it just needs a little more time to get there.

To extend your simulation, you only need to tweak a couple of elements of the simulation procedure explained above. Firstly, in your bemeta.dat files, you want to include the word “RESTART” on a line above RANDOM_EXCHANGES. This will tell your simulation to read in the previous hill files as a starting point, and to append more information to the previous files rather than begin new ones. You will also need to add onto your mdrun line the option

```
-cpi prod.cpt
```

This will take the checkpoint file outputted by your last simulation containing the position and velocities of your last frame as the start of your new trajectory. It is good, however, to start each production run in a new directory (you may need to include -noappend in your mdrun statement to get GROMACS to start new output files).

NOTE: MPI_ABORT is invoked if any files from the previous run are present in the directory of a new production run. This error can be avoided by only including .tpr and plumed files required for the extended simulation in the directory where the extended simulation is being run

Part 5: Checking that your simulation completed successfully

Here is a list of tips to check to see if your simulation completed successfully. This is by no means a proof by induction or anything to prove that it ran as intended, but is rather a nice protocol to sanity check that some major flaws did not pop up.

1. Tail the runerr file from your simulation. This is usually the first thing I do. You want to see the line “Thanx for Using GROMACS.” Otherwise, you likely had something crash partway through your simulation. You also expect your runout file to be blank.
2. Read through the log file of one of your neutral replicas and one of your biased replicas. You should also be reading the log file of your grompp process before your simulation begins. While you do not need to read every word tirelessly every time you run a simulation, it is good to do a focused skim of everything at the head and tails of the files (you can skip through the “Energies” that take up most of the file). You expect to see a reasonable performance time at the tail of your log file (I usually run 50ns/day for a 6mer with 4 cores per replica). You can then expect to roughly double or half your time by increasing or decreasing the number of cores-per-replica.
3. If you are extending your simulation, you should expect the size of your HILLS and COLVAR file to double (if you’re going from 100ns->200ns). Additionally, in the head of your biased log files, you should see a line under the section “PLUMED: Action METAD” that says that you are “Restarting from HILLS: ____ Gaussians Read”
4. Run dPCA/Cluster analysis and NIP to see that your s1 and s2 simulations have converged to a similar solution that is biophysically logical.
5. Use vmd to watch through a trajectory of one of your neutral replicas. You are mainly looking for wacky behavior - bonds being broken, cis peptide bonds being formed, etc
6. Read through your bemeta.dat files and check a few TORSION ATOMS lines to see that the atoms are correctly chosen. Also, if you’re extending your simulation, make sure your RESTART line is at the top (I made this mistake before for a bunch of simulations; it’s no fun).

7. Check your topology file to make sure that the rsff2 script was properly applied to your system and that the once “missing” improper dihedrals on the peptide bond between your first and last residues were correctly added. Also that there are no crazy behavior - a net charge in your system, correct #includes, etc. This should optimally be done before your simulation begins.