# DISCRETE MOLECULAR DYNAMICS OF PROTEINS WITH A COARSE-GRAINED MODEL

### Preparation of the input structures

The remapping of the structures from atomistic to the coarse-grained representation used by the simulation program is made with the programs **gentop** and **gentopcg**. **gentop** accepts a structure in AMBER format, and its output is the input for **gentopcg**, that generates the coarse-grained structure.

To convert any PDB file to AMBER format, you have to use **tleap**

```
tleap -f scriptleap.txt
```

Where *scriptleap.txt*, that is in the *SCRIPTS* directory

```
source leaprc.ff14SB
prot=loadpdb struct.pdb
savepdb prot leapoutput.pdb
quit
```

reads the PDB file *struct.pdb* and converts it to *leapoutput.pdb* that has AMBER format You can find lost of structures in the *structures* directory

Doing

```
gentop < leapoutput.pdb
```

the file *structure.pdb* is generated. This structure has only the heavy atoms and the hydrogens of the backbone amino group, that have been conserved in order to construct the backbone hydrogen bonds. The topology file for this structure is *topology.dat*

Finally doing

```
gentopcg < structure.pdb
```

The atomistic structure is remapped into the coarse-grained structure, *structurecg.pdb*

The topology file for this structure is *topcg.dat*

These two files are the inputs for the simulation program **cgdmd**.

### Parameters for the simulations

To run the simulation type

```
cgdmd < dmdcg.dat
```

where *dmdcg.dat* is the parameter file:

```
 &INPUT
 ,FILE9='structurecg.pdb'
 ,FILE7='topcg.dat'
 ,FILE12='energy.dat'
 ,FILE20='snapshots.pdb'
 ,FILE16='beads.dat'
 ,FILE17='potentials.dat'
 ,TSNAP=1.D-10
 ,TENE=1.D-11
 ,NBLOC=1000
 ,TEMP=300
 ,KKK=2857
 ,FVDW=8
 ,FSOLV=12
 ,ASOLV=10
 ,EPS=16.5
```

```
&END
&DISTANCIES
&END
```

The parameters are the following:

1. **FILE9**: initial structure
2. **FILE7**: topology file
3. **FILE12**: energy as a function of time (output)
4. **FILE20**: trajectory (output)
5. **FILE16**: bead library; tells which atoms are included in each bead type, and its atomtypes
6. **FILE17**: potential parameters library; tells the interaction parameters for each atomtype
7. **TSNAP**: time between snapshots, in s
8. **TENE**: time to write in the energy file, in s
9. **NBLOC**: number of snapshots of the trajectory. The total length of the trajectory is NBLOC*TSNAP
10. **TEMP**: temperature in K
11. **KKK**: random seed. To generate a different trajectory from the same initial structure, change this number.
12. **FVDW**: strength of the Van der Waals term in the coarse-grained force field
13. **FSOLV**: strength of the implicit solvent term in the coarse-grained force field
14. **ASOLV**: solvent exposition modulation parameter of the implicit solvent term

15. **EPS**: $\frac{330}{\epsilon_r}\frac{KcalA}{mole^2}$. In this example whe have chosen $\epsilon_r = 20$

The libraries *beads.dat* and *potentials.dat* are in the *lib* directory. FVDW, FSOLV, ASOLV and EPS are force field parameters that can be changed when reparametrizing the force field.

If you look at the energy ouput file (FILE12) the 1st column is the time, the 2nd column is the total potential energy (intramolecular+intermolecular), the 3rd column is the intermolecular potential energy, the 4th column is the hydrogen bond energy (intramolecular+intermolecular), and the 5th column is the intermolecular hydrogen bond energy. The last column is the number of hydrogen bonds.

## SIMULATION EXERCISES

### 1. Stability of protein complexes

Take an experimental complex structure from the directory *structures/complexes* (**ref.pdb*: complex; **lig.pdb*: ligand; **rec.pdb*: receptor). Use **tleap** as explained before to generate the structure in AMBER format. Change the format to that accepted by **cgdmd** and run the DMD simulation. Observe the trajectory with VMD and check if the complex is stable with the force field parameters used. If not, change the force field parameters.

Protein docking methods are devised to predict the correct assembly of the ligand and receptor in an experimental complex. The docking algorithms generate many different configurations of the receptor+ligand system. They use an optimized scoring function to estimate the free energy of each configuration, and rank the different configurations according to its score. Due to the difficulty of the problem, in most cases the best scored configuration (first ranked) is very different from the exprimental complex. These highly scored configurations are called false positives.

To check that the parametrization of the force field is able to distinguish between the experimental complex and a false positive, make the same simulation of the first false positive

of the same complex, the structure is in *structures/ffp*. An ideal force field would keep stable the experimental complex structures during the simulation and would dissociate the ligand and receptor assembly in the false positive configurations.

When you find a good set of parameters, please tell me! Thank you very much for your contribution to the optimization of the force field...

## 2. Aggregation of small peptides

Here you will simulate a system of several small peptides. Your aim is to find out at which concentration of peptides the system saturates and precipitates. The simulations are now made in a cubic box of side RBOX with periodic boundary conditions.

To change the concentration, you have to change the size of the simulation box, adding in the INPUT block of the parameter file *dmdcg.dat* the line

```
,RBOX=100
```

In this example a box with a side of 100 A has been defined.

Use small peptides (6 residues or less) and only 4 molecules, so the simulation will run faster. To increase the concentration, reduce the side $L$ of the cubic box ($RBOX$). Remember that the concentration (in peptides/volume) is $N/L^3$.

To construct the peptide, cut a piece of any PDB file. Keep only the CA atoms doing

```
cat structure.pdb | grep 'CA' > struct.pdb
```

And modify the residue name in the PDB file, so you can choose any sequence you want

Use **tleap** to reconstruct the structure and then change the format to that accepted by **cgdmd**. In order to generate an extended conformation, make a short run of **cgdmd** with these parameters in the INPUT block of the parameter file

```
,NBLOC=100
```

```
,FVDW=0
,ASOLV=100
,EPS=0
,RBOX=0
```

instead of those given in the example file. These parameters cancel the attractive Van der Waals term and maximize the generally repulsive implicit solvation term.

Add this line in the DISTANCIES block

```
ROHMAX=0.1
```

this tells the simulation program that the maximum distance of a hydrogen bond is 0.1 A, then the program will not form any hydrogen bond and any secondary structure of the initial conformation will disappear, leading to a random coil.

Once you have generated a conformation that you like, crop it from the *snapshots.pdb* trajectory file. Make copies of this structure with the script *translate.sh* in the *SCRIPTS* directory. Remember to label each structure with a different letter, this will allow you to visualize them better with VMD.

Join the four structures:

```
cat structure1.pdb structure2.pdb structure3.pdb structure4.pdb > structurecg.pdb
```

Check with VMD that your initial configuration of the system of four peptides that you are going to simulate is OK. VMD may not connect the backbone of your peptides when you choose a trace representation. If you want to see it, make

```
cat structurecg.pdb | grep 'CA' > structca.pdb
```

and then it will work.

To generate the topology file for this system, use the **toprep** program. Add the line

```
# 3
```

at the beginning of *topcg.dat*, that is the topology file of the peptide. We are telling **toprep** to make three repeats of the topology, since we have a PDB file with four peptides:

```
toprep < topcg.dat > topcg4.dat
```

Finally you have ready *structurecg.pdb* and *topcg4.dat* to run the simulation of the system of four peptides. Remember to put the correct parameters in the *dmdcg.dat* parameter file. Good luck!

The phenomenon to be observed is the aggregation dynamics of the system. To count the number of monomers, dimers, trimers and tetramers during the simulation, use

```
contacts < dmdcg.dat > oligomers.dat
```

This will deliver an output like this:

```
0.000   0.000  4  0  0  0  0  0
0.100   0.000  4  0  0  0  0  0
0.200   0.000  4  0  0  0  0  0
0.300   0.000  4  0  0  0  0  0
0.400   0.000  4  0  0  0  0  0
0.500   0.000  4  0  0  0  0  0
0.600   0.000  4  0  0  0  0  0
0.700   0.000  4  0  0  0  0  0
0.800   0.000  4  0  0  0  0  0
0.900   0.000  4  0  0  0  0  0
```

The first column is the time in ns and the 3rd to 6th column is the number of oligomers of 1, 2, 3 and 4 molecules. The last two columns are a flag that is different than 0 when two oligomers join (7th column) and when an oligomer dissociates (8th column). Look at this columns to find the association/dissociation events in your trajectory:

```
cat oligomers.dat | awk '{print $1,$7,$8}' | grep ' 2'
```

**contacts** generates *histolig.dat*, a histogram with the abundance of each oligomer species:

```
1   1.9100000
2  0.79500002
3  0.16666667
4   0.0000000
```

You can use **xmgrace** to visualize it.

If you find errors when using the code, please let me know. Thanks for your collaboration in the debugging of the program!