

Computational Exercise 4

One of the more difficult aspects of setting up a molecular dynamics simulation is creating a sensible starting configuration that doesn't blow up on the first few time steps. Placing molecules at random locations in a box is almost guaranteed to produce overlapping atoms and very high energy structures. The forces on these atoms will be astronomically large, and unless exceedingly small time steps are used, the simulation doesn't survive to find a more reasonable structure. To get around this, we often build "safe" structures (crystal lattices), proteins in solvated crystal or NMR-based structures, etc. and slowly relax them.

A packing optimization tool called "Packmol" automates a Monte Carlo process for placing molecules in particular geometric regions. We'll use Packmol along with some of our other tools to build a heterogeneous structure (a fatty acid bilayer). Later, we'll run a moderate length simulation of this bilayer and will analyze an orientational order parameter.

Getting the requisite molecular geometries

1. Open a browser and surf to <https://lipidmaps.org>. On the right hand side, click on the link for the "LMSD Database" and then go to "Browse Classification". We want to grab a molecule from the Fatty Acyls → Fatty Acids and Conjugates → Straight chain fatty acids group. Find the entry for Lauric acid (dodecanoic acid). It should have a Lipid MAPS id of LMFA01010012.
2. You can download a 2-D structure of this lipid by using the Download file button with the MDLMOL option selected. This should download a file called LMFA01010012.mol into your downloads directory. Once you have the correct file you can close your browser.
3. Open Avogadro, go to File → Open, and select the .mol file that you downloaded from lipidMAPS. Avogadro will ask you if you want to build a rough geometry because of the missing 3D coordinates, and you will want to answer "Yes". Set up the MMFF94 force field and choose Extensions → Optimize Geometry
4. We'll be simulating lauric acid with a united atom (UA) force field that has CH₂ beads representing each methylene group. To set things up, we first need to get rid of most of the hydrogen atoms:
 - a. Using the build (pencil) tool, turn *off* the "Adjust Hydrogens" checkbox.
 - b. Delete all of the hydrogens *except* for the alcohol hydrogen.
5. **This step is very important!** We need to have the atoms indexed in a particular order for the next step.
 - a. Go to Build → Cartesian Editor... and then do Sort by Z and hit Apply.

- b. Dismiss the Cartesian Editor window.
 - c. Turn on the Labels in the Display Settings and verify that the hydroxyl oxygen is atom 14 and the terminal carbon is atom 1. If they don't have this numbering, re-open the .mol file and start over.
 - d. Save your final structure as an XYZ file called **lauricAcid.xyz** in your home directory.
6. Open a terminal, and copy your lipid structure over to your home directory on the CRC machines. On a Mac, this is:

```
scp lauricAcid.xyz user@crcfe01.crc.nd.edu:~/chem650/
```

replace *user* with your own Notre Dame NetID. You may have to use PSCP.EXE instead of scp if you are on a Windows computer.

7. Log in to the CRC machines and load the requisite modules using slogin (or PuTTY). Some of these steps below will take some time, so if you have access to another CRC front-end node, you might want to use that instead of crcfe01:

```
slogin user@crcfe01.crc.nd.edu -Y
module load openmd
module load grace
module load vmd
cd chem650
```

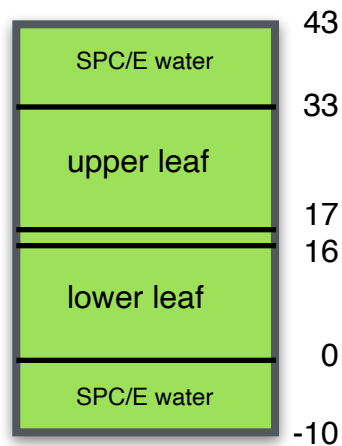
8. Next we'll copy all of the files we need to prepare a lipid bilayer:

```
cp /groups/gezelter/650/bilayer.pack ~/chem650
cp /groups/gezelter/650/water.omd ~/chem650
cp /groups/gezelter/650/freshWater.omd ~/chem650
cp /groups/gezelter/650/lauric.frc ~/chem650
```

You should now have a coordinate file (lauricAcid.xyz) in your chem650 directory as well as a packmol input file that will help prepare the configuration.

Open the bilayer.pack file in a text editor. There are a number of comments in the packmol file that explain what it does, but basically it will create a box that is 20 x 20 x 33 angstroms containing 36 lauric acid molecules in two specific regions of the box. It also will attempt to align the lauric acid molecules to point their hydroxyl ends (atom 14) at the bottom and top of the box.

We'll add water layers above and below the bilayer when we are done. In the diagram to the right, we're asking packmol to create the upper leaf and lower leaf regions with z



coordinates between 0 and 33.

```
tolerance 4.5
filetype xyz
output bilayer.xyz
```

```
structure lauricAcid.xyz
  number 18
  inside box 0. 0. 0. 20. 20. 16.
  atoms 14
    below plane 0. 0. 1. 2.
  end atoms
  atoms 1
    over plane 0. 0. 1. 12.
  end atoms
end structure
```

```
structure lauricAcid.xyz
  number 18
  inside box 0. 0. 17. 20. 20. 33.
  atoms 1
    below plane 0. 0. 1. 19.
  end atoms
  atoms 14
    over plane 0. 0. 1. 31.
  end atoms
end structure
```

9. Run the packmol program on this file:

```
packmol < bilayer.pack
```

10. Verify that the bilayer structure, `bilayer.xyz`, is sensible using `jmol` or `vmd`

11. Use the `atom2omd` program to convert the structure into a format that can be read by OpenMD:

```
atom2omd -ixyz bilayer.xyz
```

This command will create an incomplete OpenMD file called `bilayer.omd` that must be edited before it can be used. We are going to be using the united atom force field, which is described in `lauric.frc`

- a. As you have done with methanol, edit the `bilayer.omd` file and change all instances of `C3` → `CH2`

- b. Also change O3 → OH
 - c. Leave the C2 and O2 atoms alone
 - d. The CH2 atom immediately adjacent to the C2 should be atom[10]. Change the type of this atom to CH2C
 - e. The terminal methyl atom (atom[0]) needs to be called a CH3 atom, so change it from CH2 to CH3
 - f. Change the name of the molecule to something descriptive (use “lauricAcid”). This should be done in two places; once in the molecule description and another time in the component block.
12. Because we’ll be doing some more complicated work on our input files, use your text editor to create a *new* file called `lauricAcid.omd` and copy the 34 lines of the molecule definition into this file.
13. In `bilayer.omd`, you should now replace the molecule definition (all 34 lines) with a single statement:

```
#include "lauricAcid.omd"
```

14. **This step is very important!!** Packmol places molecules right up to the boundaries, which are adjacent to each other in periodic boundary conditions. We need to give the molecules a bit of a buffer between periodic boxes, so find the Hmat line and enlarge the x, y, and z-axes of the box so that we’ll also have room to include our water. The final Hmat line should look like:

```
Hmat: {{ 23, 0, 0 }, { 0, 23, 0 }, { 0, 0, 53 }}
```

15. Before the simulation can run, add a `forceField` line after the `component` block:

```
forceField = "lauric";
```

At this stage, you should be able to run OpenMD on the file to check to make sure your hand-crafted atom typing can be matched up with types known by the force field:

```
openmd bilayer.omd
```

You can safely ignore any warnings about inversion types and the “IntegratorFactory”.

The Water Box

16. We have supplied you with a box containing 1100 SPC/E water molecules of the same dimensions as your bilayer box. This file is called `freshWater.omd`, and you can visualize what’s in it:

```
Dump2XYZ -m -b -i freshWater.omd  
jmol freshWater.xyz
```

Combining the Bilayer and Water boxes into one simulation

17. Next, we'll take the `bilayer.omd` file and the `freshWater.omd`, and combine them together, removing any overlapping water molecules within 4.3 angstroms of any of the lauric acid atoms. We also need to specify how many objects there are in each of the two molecules, and for united-atom lauric acid, this number is 15. For Rigid Body water molecules, the number is 1:

```
omd-solvator -u bilayer.omd -v freshWater.omd -r 4.3 -o combined.omd -n 15 -p 1
```

18. When this is complete, the combined omd file should only have around 270 water molecules remaining, but it has been stripped of the molecule definitions. To use the include files, you want to add these lines *above* the component blocks:

```
#include "water.omd"  
#include "lauricAcid.omd"
```

19. Add the following lines *below* the component blocks of the `combined.omd` file:

```
forceField = "lauric";  
ensemble = NVT;  
cutoffMethod = "shifted_force";  
electrostaticScreeningMethod = "damped";  
cutoffRadius = 9;  
dampingAlpha = 0.2;  
targetTemp = 300;  
tauThermostat = 1000;  
dt = 1.0;  
runTime = 1e3;  
sampleTime = 100;  
statusTime = 10;
```

20. Initial configurations that are created from bare structures typically have no velocity information. To give an initial kick to the atoms (i.e. to sample the velocities from a Maxwell-Boltzmann distribution), you can use the following command:

```
thermalizer -o warm.omd -t 300 combined.omd
```

This creates a new OpenMD file called `warm.omd` which has initial velocity information.

21. At this stage, a simple simulation can be run:

```
openmd warm.omd
```

22. This should complete relatively quickly, and should create four new files:

- `warm.stat` (reports the status of the simulation as it progresses)
- `warm.dump` (contains the actual trajectory data)
- `warm.eor` (an “end-of-run” file that contains the single most recent stored configuration)
- `warm.report` (a file that contains statistical averages of the quantities in the stat file)

23. To view the contents of the trajectory file, you’ll need to convert the dump file into something another program can visualize:

```
Dump2XYZ -i warm.dump -b -m --repeatX=1 --repeatY=1 --repeatZ=1
```

This will create a new file called `warm.xyz` which replicates the contents of the box once in each direction. To visualize the trajectory, you can use:

```
vmd warm.xyz
```

24. The “End-of-Run” file, `warm.eor` can be re-purposed as the starting point for a new simulation,

```
cp warm.eor stable.omd
```

Edit the `stable.mod` file, and change parameters you’d like to change before running `openmd` on the new file. Pick a run time of around 50000 fs (e.g. 50ps), and a sample time of 500 fs. This will use up quite a bit of CPU time, so if you are on a node with a some free processors, you can run this job using more processors:

```
mpirun -np 4 openmd_MPI stable.omd
```

25. Collect some data!

Analyzing the results: S_{CD} order parameter

The S_{CD} order parameter is often reported in the experimental characterizations of lipid bilayers. It is obtained through deuterium NMR, and measures the ordering of the carbon-deuterium bond in relation to the bilayer normal at various points along the chains. The order parameter has a range of $[1, -1/2]$. A value of 1 implies that the CD bond is ordered and is lined up parallel to the bilayer normal vector, while 0 implies full disorder, and $-1/2$ implies full order *perpendicular* to the bilayer normal. The CD bond vector for carbons near the head group are usually ordered perpendicular to the bilayer normal, with tails farther away tending toward disorder. This makes the order parameter negative for most carbons, and so $|S_{CD}|$ is more commonly reported than S_{CD} .

In our model, there are *no* explicit hydrogens, but the order parameter can be deduced by using the relative positions of the carbons at each point in the chain:

$$S_{CD} = \frac{2}{3}S_{xx} + \frac{1}{3}S_{yy}$$

where

$$S_{ij} = \frac{1}{2} \left\langle (3 \cos \Theta_i \cos \Theta_j - \delta_{ij}) \right\rangle.$$

Here, Θ_i is the angle between the bilayer normal (the z axis of the box) and the i^{th} vector in the reference frame of the carbon. The brackets indicate an average over time and molecules. The reference frame for carbon n is defined in terms of the neighboring carbons in the chain:

z : vector from C_{n-1} to C_{n+1}
 y : vector perpendicular to z, and in the plane containing $C_{n-1} - C_n - C_{n+1}$
 x : a vector that is perpendicular to y and z

In coarse-grained models, even though there are no explicit hydrogen atoms, the order parameter can still be written in terms of carbon ordering at each point of the chain.

1. First analyze the trajectory (dump) file to find S_{CD} as a function of position along the chain:

```
StaticProps -i stable.dump -s --molname="lauricAcid"
             --begin=0 --end=11
```

This command should be all on one line. This computes S_{CD} for a sequence of atoms starting at the tail CH3 (atom[0]) and moving up to the carboxylic C2 carbon at atom[11]. Because we need to use atom[0] to compute the S_{CD} for atom[1] and atom[11] to compute S_{CD} for atom[10] there will only be ten values in the data file:

```
more stable.scd
```

2. What does the S_{CD} data tell you about order in the bilayer? If you run for longer, do the results change?

Analyzing the results: Electron Density Profile

The electron density profile is proportional to the density profiles measured by x-ray scattering experiments,

$$\rho_{\text{x-ray}}(z) \propto \sum_{i=1}^N \frac{n_i}{V} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(z-z_i)^2/2\sigma^2}$$

where n_i is the atomic number of atom i , σ is the spatial extent of the atom's electron cloud (approximated by the Lennard-Jones σ), and z_i is the z coordinate of that atom.

1. First analyze the trajectory (dump) file to find the electron density as a function of the z -coordinate:

```
StaticProps -i stable.dump -d --sele1="select lauricAcid"
            --len=53 --nbins=200
```

this command should be all on one line. View what the output looks like:

```
xmgrace stable.density
```

2. Next try the same thing, but just for the C2 atoms:

```
StaticProps -i stable.dump -d --sele1="select C2"
            --len=53 --nbins=200
```

3. Can you think of a better way of doing the electron density than using the atomic number? What other information do we have about these atoms?
4. Find the diffusion constant for the C2 atoms. Do the C2 atoms diffuse in 2 or 3 dimensions? Now do it for the SPCE molecules.
5. Turn in all of your plots, a discussion of your S_{CD} data, and any suggestions for improving electron density calculations.