Optimal **M**embrane **G**enerator
(**w**ith **t**emplate **f**iles)

The LOOS membrane builder
version 1.0

Dr. Alan Grossfield: alan_grossfield@urmc.rochester.edu

Contents

1 Introduction

Welcome to the **O**ptimal **M**embrane **G**enerator **w**ith **t**emplate **f**iles (**OMG**, for short), the LOOS membrane building tool. The goal of this suite is to facilitate the construction of complex systems containing lipid bilayers and integral membrane proteins, as well as soluble systems (using solvate.py). The approach used here merges two previously found in the literature: the "library of conformations" approach pushed by Grossfield et al. (PNAS, 2006, 103, 4888-4893) and the INFLATEGRO strategy developed by Tieleman and others. The basic algorithm is to start with libraries of previously generated conformations for the lipids used (generally, one uses neat simulations to make the libraries, but one can use model building as well to create an initial library), and pack them around the "protein", in any. The system's periodic box is initially scaled to 3x its original size in the plane of the membrane (the normal is always set parallel to the z-axis), allowing the lipids to be randomly placed with relatively few collisions. We start by placing the protein (if any) at the origin. Treating the two leaflets independently, we sequentially add lipids by picking a conformation from the library, randomly placing it in the appropriate leaflet and checking for collisions with previously placed lipids. This continues until all of the lipids are placed. Next, the system is subjected to a series of energy minimizations, each followed by a small scaling of the positions of the molecules in the plane of the membrane, gradually shrinking the membrane down to its target size.

OMG supports the creation of arbitrarily complex membrane compositions, including mixtures of lipids and asymmetric bilayers. The user simply creates a configuration file that specifies the segments to be included and which leaflet they should be found in, as well as the number of ions and water molecules.

2 Requirements and Installation

To use the membrane builder, one must first have a working LOOS installation, with PyLoos working. If you have not downloaded LOOS yet, you can download it from `https://github.com/GrossfieldLab/loos`, and follow the instructions included there for building.

You'll also need to make sure your paths are set up correctly, so that python can find the correct files. The LOOS and PyLoos parts of this are best handled by sourcing the included files `setup.csh` or `setup.sh` (for `t/csh` and `bash`, respectively). In addition, you'll need to append the directory where the python files from this code are installed to your `PYTHONPATH`.

You will need a set of lipid template files. These are sets of lipid conformations, each in the its own coordinate file, which are used in the construction of new lipid membranes. We provide a preliminary set for some of the more common lipids as a separate download, available from `http://membrane.urmc.rochester.edu/sites/default/files/lipid_library.tgz`. Section 5 describes how you could go about making your own library of lipid conformations.

Finally, you will need a working NAMD install, including NAMD itself and `psfgen`. At this point, we assume you have the ability to run NAMD directly on the local machine, rather than via a queuing system. The jobs shouldn't take very long (less than 20 minutes for the examples I've done).

3 Usage

OMG is controlled by a configuration file, given as the first command line argument; a second optional command line argument specifies the output directory, where all files will be created (it will use `./out/` if nothing is specified). Everything that is needed, including the location of the topology and parameter files, the box size, the number of each lipid type, and the location of the structure libraries are all declared here.

We assume that lipids are laid out in the 1 residue/lipid format used in the CHARMM36 parameter files, as opposed to the earlier 3 residue/lipid (chain-headgroup-chain) format often used with CHARMM27. Furthermore, we adopt the convention that each lipid type in each leaflet will be its own segment. So, a pure POPC bilayer would have two segments, one for the upper leaflet and one for the lower. While this doesn't affect the running of the simulation in any way, it can greatly simplify the analysis by giving an easy way to select specific leaflets without reference to the coordinates.

The final optimized structure will be `final.pdb`, created in the output directory.

If you are building a system that contains a protein (or protein-like molecule), you will need to start out with a working PSF and PDB file for it; there are too many potential subtleties in creating a protein (disulfides, unusual charge states, etc) for me to try to programmatically generate the PSF for you. The PSF you supply will be included in the psfgen script **OMG** creates to build the whole system.

While running, **OMG** will write a series of comments to `STDERR`; most of these are simply updates to let you know where it is in the process, or error messages if it has to stop; it will also issue a warning if the final system has a net charge, since this is usually not intended. `STDERR` will also contain output from `psfgen`, which can get somewhat long. I'd suggest a quick glance through the output the first few times you use **OMG**, even if things seem to have worked correctly, just in case.

In principle, one could take the `final.pdb` generated by **OMG** and begin production calculations, presumably in the NPT ensemble. However, we suggest running a short ($<$ 10 ns) run in the NPAT (constant area) ensemble first; depending on how many waters need to be deleted to get to the target value, the density in the bulk region can be a bit low, and NPAT can be a bit more stable when adjusting the overall volume.

Building a non-membrane system works much the same way, except instead of running **OMG** you run **solvate.py**

3.1 Format of the configuration file

The config file is key:value based. Keys are case-insensitive while values are case-sensitive. Keys are assumed to start in the first column of the line. Blank lines and lines starting with "#" are ignored as comments. Unrecognized keys will generate a warning but will not cause an error.

The following keys are used:

- **topology filename**: **filename** is the path to the NAMD-style topology file. REQUIRED
- **parameters filename**: **filename** is the path to the NAMD-style parameter file. REQUIRED
- **psf filename**: **filename** is the path to the NAMD-style psf file that will be generated by **OMG**. Typically given as a raw filename without a path. REQUIRED
- **segment segname resname num-lip phos-pos phos-name placement library**:

  - **segname** is the name of the segment
  - **resname** is the residue name of the lipid, e.g. POPC
  - **num-lip** is the number of lipids in this segment
  - **phos-pos** is the average distance of the phosphorus atom (or whatever atom is used for centering) from the membrane center
  - **phos-name** is the atom used to place the lipid; the obvious choice is "P", for the phosphorus, but you could use a different interfacial atom to place molecules that don't have phosphorus, e.g. cholesterol
  - **placement** controls the location of the lipid. "1" indicates the $+z$ leaflet, "-1" indicates the lower leaflet, and "0" indicates a transmembrane species
  - **library** is the directory containing a library of conformations for this lipid

  At least 2 segments required (one for each leaflet)

- **water segname resname num-water thickness box-size coords**:
  - **segname** is the name of the water segment. For now, we assume that you have fewer than 100,000 waters, so they can all be put into 1 segment. We do not support multiple water segments.
  - **resname** is the residue name of the water, e.g. TIP3
  - **num-water** is the number of water molecules to include
  - **thickness** the z dimension of the box of water to be generated (the x and y dimensions are inherited from the box specified for the system as a whole)
  - **box-size** is the dimension of the water box named by the **coords** keyword (presumed to be a cube)
  - **coords** is a file of coordinates for a pre-equilibrated box of water, which will be tesselated to create a box of the appropriate size. This could be a PDB file or a CHARMM-style CRD file (`water_small.crd` is included in the distribution). Probably any file format supported by LOOS would work here, but only those have been tested.

REQUIRED

- **salt segname resname number**:
  - **segname** is the segment to assign this ion to
  - **resname** is the residue name of the ion, e.g. SOD. At the moment, this is presumed to be a single-atom ion, as opposed to something more complex like SO_4.
  - **number** is the number of this ion to include

OPTIONAL

- **protein model-file psf-file water-segment scale-by-molecule**
  - **model-file** is a file specifying the contents of the "protein". This is presumed to be a PDB file, though other structure formats supported by LOOS should work as well. The program does not translate or reorient this molecule; you should have it located as you want it to be in the final structure. The "protein" can have as many segments as you want, and can contain as many molecules as you want (e.g. a protein plus ligand and palmitoylation sites). If you don't have a water segment, use "NONE" or some non-existent segmentname for the water segment.
  - **psf-file** is a NAMD-style psf file to create the protein. This will be used to make the other psf files needed to run NAMD during the build process.
  - **water-segment** is the name of a segment in the model-file that contains water molecules. The coordinates of these waters will be retained, but near the end of construction they will become part of the segid specified in the **water** line above. If you have no water segment, put "NONE".
  - **scale-by-molecule** determines whether the molecules specified by protein get scaled outward in x and y the way lipids do. If you're working with a protein, you probably want this set to "0", for false. However, if your protein is actually a set of independent molecules (e.g. a group of peptide scattered on the surface), you would set this to "1", indicating true.

OPTIONAL

- **box x y z**: **x, y, and z** are the target dimensions of the system periodic box. REQUIRED
- **namd namd-binary**: path to the binary for NAMD. OPTIONAL: defaults to `/opt/bin/namd2`
- **psfgen psfgen-binary**: path to the binary for `psfgen`. OPTIONAL: defaults to `/opt/bin/psfgen`

For a soluble system, you'd use **solvate.py**, but the basic options are the same. You wouldn't have any **segment** lines, since those specify lipids. In addition, there's one more option avaiable:

protrot VALUE: setting this option applies a random rotation to the protein, to ensure that each time the calculation is run, you get a different solvation set up. **VALUE** should be any non-zero value.

3.2 Examples

Below are 3 example configuration files, in ascending level of complexity. Please note that these are demos only, and that I've made no effort to include correct box sizes or salt concentrations. They are provided to give working examples for how to run **OMG**. The equivalents of these files are found in the `example` directory of this distribution; you will need to update the paths to reflect your particular filesystem, but otherwise they should work for you out of the box.

To run one of these examples, cd into the appropriate directory (e.g. `example/popc`), and run the **OMG** giving the config file on the command line. For example, the POPC example below would be run as

`../../OptimalMembraneGenerator.py popc.cfg test`"

This would run the **OMG** and put the output in the directory `test`. On my computer, the examples take about 10 minutes to run (your mileage may vary).

Note: if you haven't downloaded a lipid library yet, now would be a really good time to do so. You can get one from `http://membrane.urmc.rochester.edu/sites/default/files/lipid_library.tgz`.

3.2.1 Pure POPC bilayer

```
# Location of the topology and parameter files
topology /home/alan/projects/loos-membranes/toppar/top_build.inp
parameters /home/alan/projects/loos-membranes/toppar/par_build.inp

# Name of the output psf file
psf popc.psf

# Size of the final system periodic box
box 75.9 75.9 78.5

# Segment for the upper leaflet
segment TPC POPC 60 19 P 1 /opt/lipid_library/popc_c36
# Segment for the lower leaflet
segment BPC POPC 60 19 P -1 /opt/lipid_library/popc_c36

# Water and salt
# Note: line-break in the "water" line is just in this document
water BULK TIP3 8000 50 15.5516 /home/alan/projects/loos-
membranes/water_small.crd
salt SOD SOD 10
salt CLA CLA 10
```

3.2.2 Mixture of POPE and POPG

```
topology /home/alan/projects/loos-membranes/toppar/top_build.inp
parameters /home/alan/projects/loos-membranes/toppar/par_build.inp
psf pe_pg.psf
box 75.5 75.5 78.5

# Upper leaflet
segment TPE POPE 60 19 P 1 /opt/lipid_library/pope_c36
segment TPG POPG 30 19 P 1 /opt/lipid_library/popg_c36
```

```
# Lower leaflet
segment BPE POPE 60 19 P -1 /opt/lipid_library/pope_c36
segment BPG POPG 30 19 P -1 /opt/lipid_library/popg_c36

# Salt and water. Note extra SOD to provide neutrality
# Note: line-break in the “water” line is just in this document
water BULK TIP3 8000 50 15.5516 /home/alan/projects/loos-
membranes/water_small.crd
salt SOD SOD 70
salt CLA CLA 10
```

3.2.3 Rhodopsin in a mixed bilayer

```
topology /home/alan/projects/loos-membranes/toppar/top_build.inp
parameters /home/alan/projects/loos-membranes/toppar/par_build.inp
psf rhod.psf
box 75.5 75.5 78.5

# Starting structure for the protein. Contains multiple segments,
# including ”WAT”, which is water
protein all-rhod.pdb all-rhod.psf WAT 0
# Segname Resname number 1/-1 (for upper leaflet, lower leaflet)
segment TPE POPE 60 19 P 1 /opt/lipid_library/pope_c36
segment TPG POPG 30 19 P 1 /opt/lipid_library/popg_c36
segment BPE POPE 60 19 P -1 /opt/lipid_library/pope_c36
segment BPG POPG 30 19 P -1 /opt/lipid_library/popg_c36

# Note: line-break in the “water” line is just in this document
water BULK TIP3 8000 50 15.5516 /home/alan/projects/loos-
membranes/water_small.crd
salt SOD SOD 70
salt CLA CLA 10
```

3.3 Algorithm

The main body of the **OMG** is `OptimalMembraneGenerator.py`. As provided, it is written as a single large function, with a series of steps to build a membrane. Given how different the additional requirements could be (multiple bound proteins, peptides in the water region, domains etc), I believe that attempting to account for every possibility would produce a program so complex and cumbersome that it would be too hard to use. Instead, I'm choosing to produce a relatively simple program that can be customized to solve each individual problem.

We start with the configuration file (described above in Section 3.1) and the lipid template files (see Section 5). The program starts by looping over the segments listed in the configuration file. For each segment (presumably a lipid species for a particular leaflet), we pick a random configuration from the library, place it at the origin, and apply a random rotation about the z axis. If it's going to be in the lower leaflet, it is flipped upside down via a 180° rotation about the x-axis. The molecules is then shifted so phophorus atom is then placed at the origin, and then molecule is shifted to its new position. The x and y coordinates are generated randomly within a box 3 times larger than the final system target size, while the z position of the phosphorus is set to be the phosphorus height set in the config file, $\pm 0.5 \AA$ generated randomly; if the lipid is targeted to the lower leaflet, then this value is negated.

Before the new lipid placement is accepted, we perform a series of bump checks, first against the protein (if one was specified in the config file), then against previously placed lipids; the configuration is rejected its placement creates at least 3 collisions (defined as a pair of atoms within 3 Å of each other). Once all lipids have been placed, we write out a PDB file of the "scaled" coordinates, called "lipid_only.pdb".

Next, we perform a series of scalings and minimizations. First, the center of mass of each individual molecule is scaled downward in the xy plane. The new coordinates are written out as a PDB file called "lipid_shrink_i.pdb" (where "i" is the iteration number), in the working directory specified on the command line. This PDB file is then used to run a 100 step minimization in NAMD, with 10 kcal/mol-Å^2 restraints applied to each heavy atom only in the z direction; the purpose of the restraints is to make sure that early collisions don't lead the membrane to buckle unphysically, instead making lateral displacements more favorable. The resulting coordinates are then read back in to the **OMG**. After recentering the entire membrane at the origin, the process is repeated until the system reaches the target size.

Next, we build a box of water using the provided template box, by replicating the box in each dimension such that its large enough to cover the required volume (the x and y coordinates match the system ones, while the z dimension is calculated by subtracting 29 Å from the system z dimension. This is currently hardwired in, and probably should be replaced by something based on the positioning of the phosphates. In any event, some system-specific trial and error is often necessary to get the overall density right.

At this point, the number of water molecules should be significantly larger than the target value. We then superpose the water box onto the previously placed lipids and protein, and remove waters that clash, where a clash is defined as a contact between a water oxygen and any heavy atom closer than 1.75 Å. We verify that we still have enough water, and then proceed to add the salt ions (if any were requested in the config file) by randomly selecting and replacing water molecules.

If at this point we still have too many waters, we then proceed to randomly delete waters to get down to the target value. Once this is accomplished, we then merge the waters (if any) specified as part of the protein (if any) into the water segment (called "BULK" in most of our examples).

We construct the final system, with the atoms in the following order: protein (if any), lipids, salt, then finally water. We write out the coordinates in the working directory, in a file called `final.pdb`. We also write a psfgen script that will generate the full PSF file for the system, called "generate_full_psf.inp", and run it for you, creating a PSF with the output name you specified in the config file.

4 Adding small molecules to a protein simulation

If you need to add a bunch of small molecules to a simulation (e.g. ligand molecules that you want to allow to bind to a protein), you can use the **add_molecules.py** tool to do it. It uses the same kind of library-based approach as the rest of the **OMG**, although there is no config file. Instead, it takes command line options only.

The mandatory command line options are:
`add_molecules.py num_ligands box_size library_location output_pdb`
where:

- **num_ligands**: number of ligand molecules to add
- **box_size**: size of cubic box, in Å
- **library_location**: directory where the molecule library is found
- **output_pdb**: resulting pdb file

Additionally, there are a number of options one can specify:

- **protein**: coordinate file for your protein (can be anything)
- **no_center**: do not center the protein coordinates at the origin
- **zbox**: set the z-dimension of the box, if different from xy
- **z_exclude**: don't place molecules within this distance of $z = 0$

5 Lipid Template Files

The lipid template files are libraries of conformations, used to seed the membranes with a realistic ensemble of structures. The "better" the ensemble of structures used — meaning, the closer to that sampled by the real liquid crystalline membrane — the less equilibration will be required before the resulting simulation data is useful. For this reason, the templates distributed here are the result of extensive simulations, some performed with applied tension such that the area/lipid and ^{2}H NMR order parameters match up with experiment.

5.1 Templates included in the distribution

We have included libraries for commonly used lipids. Or rather, for lipids commonly used by our group. In all cases, the lipids use CHARMM36 parameters and atom naming (although not all of the lipids are explicitly in the released CHARMM36 files):

- POPC

- POPE
- POPG
- SDPC
- SDPE

5.2 Making your own templates

If you're creating a library for a new lipid, you may or may not have a neat simulation to use to build the library. In that case, your best bet is probably to proceed via 2 steps. First, "fake" a library some other way, by modifying an existing one. For example, our original library for POPE was built from the POPC library, manually replacing the choline methyls with hydrogens. Similarly, one could imagine adding or removing carbons from the acyl chains. The resulting libraries won't be very good, but one could then use them to build a few neat bilayers; after running them long enough to equilibrate them (several hundred nanoseconds may be necessary), the resulting trajectories can be used to create a new, better library.

Each library file contains a single lipid molecule, with the phosphorus at the origin and the headgroups pointing in the $+z$ direction. Other than these translations, the coordinates should be exactly as extracted from the simulations, so that the correct overall tilt is retained, and there is no imposed orientation (e.g. the glycerol backbone isn't aligned along the x-axis).

To make the process of building a lipid simpler, we provide **make_library.py**. This takes a lipid simulation, extracts the lipids, and writes each lipid from each trajectory snapshot into its own numbered pdb file.