

Ceramide — VDAC1 binding in a membrane

Note 1: This tutorial relies on the data from this [archive](#). When you unpack it, you'll find two directories. Directory `worked` has all the intermediate steps and results, whereas directory `minimal` has only the essential files to get you going, and you'll have to do most of the work yourself. Feel free to follow the tutorial using either approach, or even a mix of the two.

In directory `worked` you'll also find a `workflow.sh` script that can essentially run most of the steps in this tutorial, as well as the VMD visualization scripts used in image generation.

Note 2: This tutorial relies on some dependencies: python packages `vermouth`, `numpy` and `matplotlib` (install with `pip install vermouth numpy matplotlib`), and DSSP for secondary structure identification (on Ubuntu, install with `sudo apt install dssp`)

1. Introduction

The role of mitochondria in apoptosis is well established, and a number of protein–protein interactions in its outer membrane has been linked to apoptotic pathways. The VDAC (voltage-dependent anionic channels) family of proteins is central to this binding, having been implicated in the retention of apoptotic proteins in the outer mitochondrial membrane¹ and self-oligomerization² — an event also linked to apoptosis.

In a recent study we have ascertained that ceramide binds VDAC proteins (via a binding site involving VDAC1's Glu73) and that such binding somehow triggers apoptosis³. An immediate hypothesis is that the presence of ceramide affects the binding profile of VDAC and promotes apoptosis-associated oligomerizations.

Since that work was done using the (now outdated) Martini 2 model, we will recreate in this tutorial the same ceramide binding simulations, but using the state-of-the-art Martini 3 model⁴.

1 – *Sci. Rep.* 6, 32994; doi: 10.1038/srep32994 (2016)

2 – *Mol. Cell. Biol.* 30, 5698; doi: 10.1128/MCB.00165-10 (2010)

3 – *Nat. Commun.* 10, 1832; doi: 10.1038/s41467-019-09654-4 (2019)

4 – *Nat. Methods* 18, 382; doi: 10.1038/s41592-021-01098-3 (2021)

2. System preparation

2.1. Coarse-graining a protein with Martinize2

In your directories you'll find the `4c69.pdb` file, containing the X-ray structure of the murine VDAC1 protein. Have a look at it using your preferred structure visualization tool. As it's a beta-barrel, it should be clear how it orients in a membrane.

For CG MD simulation of a protein one needs the CG coordinates and the CG topology. Both can be obtained with the `martinize2` tool (despite its name, it works with Martini 3). Run it with

<code>./martinize2 -f 4c69.pdb \</code>	The atomistic input file
<code>-x VDACL_cg.pdb \</code>	The output CG structure
<code>-o topol.top \</code>	The output topology header file
<code>-ff martini3001 \</code>	The force field version to use
<code>-elastic \</code>	Elastic networks to restrain secondary structure
<code>-scfix \</code>	Fixes excessively flexible side-chain behavior
<code>-cys auto \</code>	Automatically detects disulfide bridges
<code>-dssp dssp \</code>	The program to classify secondary structure
<code>-ignore HOH ATP MC3 LDA \</code>	Residues to ignore in the .pdb
<code>-maxwarn 41</code>	Ignore these many warnings (in our case, these alert for the existence of alternative conformations)

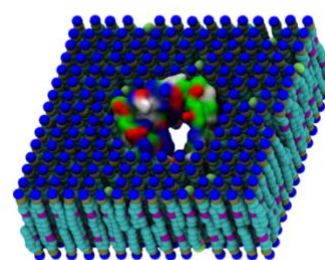
Have a look at the produced files with an editor or a molecular structure visualizer.

2.2. Embedding a protein in a membrane using *insane*

The `insane.py` script creates CG membranes by placing lipids in a grid. It can also embed proteins as it constructs a membrane. Insane leverages the robustness of CG to start from artificial conditions and eventually equilibrate. Run it with

<code>./insane.py -f VDACL_cg.pdb \</code>	The input protein to embed
<code>-l POPC:95 -l DPCE:5 \</code>	The lipid proportions (in our case, 95:5 POPC:ceramide)
<code>-x 12 -y 12 -z 10 \</code>	The output structure dimensions
<code>-o membrane_prot.gro \</code>	The output structure
<code>-p membrane.top \</code>	The output topology header
<code>-center \</code>	To vertically center the protein in the membrane
<code>-sol W \</code>	To add water as the solute
<code>-orient \</code>	To orient the protein tilt in the membrane
<code>-salt 0.15 \</code>	The ionic strength to add
<code>-charge auto</code>	To neutralize the system with additional counterions

You should obtain a protein–membrane structure like this one. You may notice that despite the use of the `-orient` flag, *insane* was not able to completely align VDAC's axis with the membrane normal. This is ok, again because the robustness of CG will allow the system to quickly evolve into the correct orientation.



2.3. Cleaning up run-files

At this point you'll have the structure needed to run simulations, but the topologies/topology headers need some cleaning:

1. *Martinize2* gives not very informative default names to files and molecules. Rename `molecule_0.itp` to `VDACL.itp`. Inside it, also change the molecule name from `molecule_0` to `VDACL`.
2. In `topol.top`:
 - a. add the last 7 lines from `membrane.top`. These are the number of molecules of each type created by *insane*.

- b. change the molecule reference from `molecule_0` to `VDAC1`;
- c. at the top, change the `#include` statement so it points to `martini_v3.0.0.itp` (this `.itp` must always be the first one to include);
- d. add extra includes for the lipid, ceramide, water and ions:


```
#include "martini_v3.0.0_phospholipids_v1.itp"
#include "DPCE.itp"
#include "martini_v3.0.0_solvents_v1.itp"
#include "martini_v3.0.0_ions_v1.itp"
```
- e. change the ion names from `NA+/CL-` to `NA` and `CL`, respectively (insane uses the older Martini 2 nomenclature).

Note: This tutorial uses a Martini 3 ceramide topology that is simply an adaptation of the Martini 2 one, with adjusted bead types. A more in-depth reparameterization is likely needed to fully leverage the accuracy of Martini 3.

3. Simulation

You can now run the simulations. In CG it is enough to perform a short energy minimization and a single pressure/temperature equilibration step before production.

3.1. Energy-minimization

The run file `em.mdp` has the instructions for performing the energy minimization. Run

```
gmx grompp -f em.mdp -p topol.top -c membrane_prot.gro -o em.tpr -maxwarn 1
```

to create an `em.tpr` processed run binary. The `-maxwarn 1` flag tells `grompp` to ignore a warning that crops up because of unimportant mismatches in the way insane names atoms and how they're defined in the topologies.

Run the minimization with

```
gmx mdrun -deffnm em -v
```

3.2. Equilibration

The equilibration step makes use of the `eq.mdp` runfile. Prior to using it with `grompp`, you'll have to create an index file defining the beads that are part of the protein/membrane and those that are part of the solvent. This is needed because the temperature of the two phases will be thermostatted independently and we need to tell `grompp` which atoms get temperature-coupled together. To make the index run

```
gmx make_ndx -f em.tpr
```

This drops you into an interactive selection interface. Input the following selections:

```
"W" | "ION"      All beads in either the W or the ION groups;
name 17 Solvent   Rename the above group to Solvent;
! 17              Invert that group, to obtain the bilayer;
name 18 Bilayer   Rename the above group to Bilayer;
```

- a BB Create a group with only the protein's backbone beads (needed later for trajectory fitting);
- q Save (by default to `index.ndx`) and exit.

Armed with this index and the minimized structure you can run

```
gmx grompp -f eq.mdp -p topol.top -c em.gro -n index.ndx -o eq.tpr -maxwarn 1
gmx mdrun -deffnm eq -v
```

Finally, run the production run, using the `md.mdp` runfile

```
gmx grompp -f md.mdp -p topol.top -c eq.gro -n index.ndx -o md.tpr
gmx mdrun -deffnm md -v
```

Note: To lighten the size of the archive, only the last processed and fit trajectory is included. See section 4.1 for details on fitting.

4. Analysis

4.1. Trajectory fitting

It is useful — mostly for visualization — to have all the protein in all simulation frames centered and fit, so that all observed movement is relative to it, rather than to the box. At the same time we will output only the lipids/protein to make the resulting trajectories lighter. This can be done with the following series of commands:

centers and makes molecules whole; select 'Protein' for centering and 'Bilayer' for output.

```
gmx trjconv -f md.xtc -s md.tpr \
           -center -pbc mol \
           -o md_pbc.xtc -n index.ndx
```

same as above but on the starting structure of the production run, to produce one centered structure; useful later for use with VMD.

```
gmx trjconv -f eq.gro -s md.tpr \
           -center -pbc mol -o md_pbc.gro -n index.ndx
```

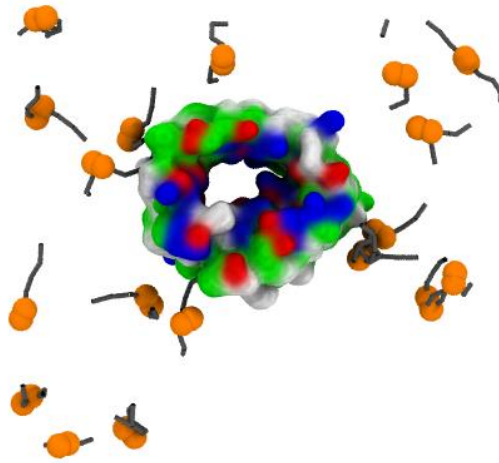
converts the production tpr file into another with fewer atoms compatible with our centered trajectory (since we now ignore the solvent); select group 'Bilayer' for output.

```
gmx convert-tpr -s md.tpr -n index.ndx -o fit.tpr -nsteps -1
```

Fits each frame's protein to the conformation in fit.tpr, by aligning backbone beads; select group 'BB' for fitting and group 'Bilayer' for output.

```
gmx trjconv -f md_pbc.xtc -s fit.tpr \
           -fit rotxy+transxy -o md_fit.xtc \
           -n index.ndx
```

Take a look at the resulting trajectory with VMD — possibly showing only VDAC1 and the ceramides — and see how trajectory fitting highlights protein–ceramide interactions (see section 4.3 for visualization tips and advanced analyses using VMD).



4.2. Ceramide contact counting

To count contacts we first get a list of closest distances between any ceramide and each residue of the protein, over time, using `gmx mindist`.

res_dists.xvg will hold a set of distances — as many as protein residues — for each trajectory frame; select 'Protein' as the first group and 'DPCE' as the second.

```
gmx mindist -respertime -s fit.tpr -f md_fit.xtc -or res_dists.xvg
```

The included script `contact_fraction.py` converts these distances into contacts, and represents them as the fraction of the total trajectory time during which a given residue was in contact with a ceramide. It assumes a 6 Å cutoff for considering contacts. Run it like this:

the script outputs a `contact_fraction.pdf` plot of the fraction of time in contact as a function of residue number.

```
python3 contact_fraction.py
```

The script's output already shows some regions of preferred contact, but it is not so clear among all the contacts whether there is a binding site (ceramides do seem to also bind on the region around Glu73, which is encouraging). In the next section we perform a more visual contact analysis.

4.3. Visualization

VMD is a great tool to visualize CG trajectories. As a general tip, white background, no fog, and no perspective often make for clearer images.

Load the fitted .gro and trajectory and set up the following selections/representations:

```
name BB "SC."
```

This selects the protein beads; use QuickSurf representation and ResType coloring.

```
resid 73
```

This selects the known binding Glu73 residue; use a large VDW representation in red, to highlight this spot.

```
resname DPCE and name "AM."
```

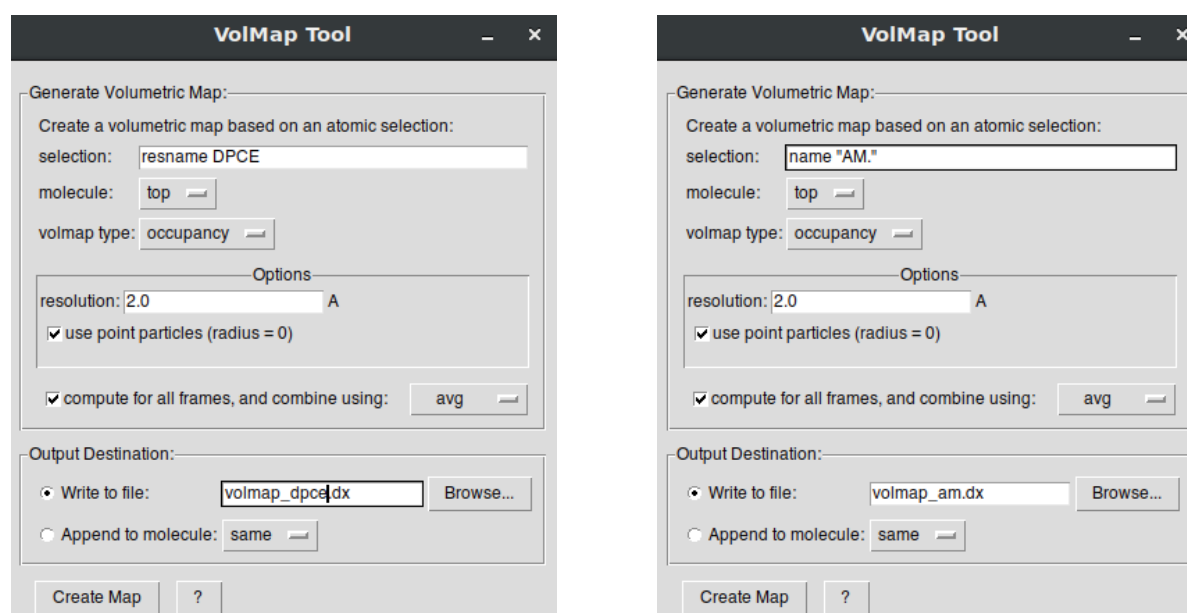
This selects the ceramide amide beads only; represent as VDW spheres;

```
resname DPCE
```

This selects all the ceramides, including AM beads; choose a dynamic bonds representation, with cutoff around 5.1Å; this way VMD will draw most of the ceramide tail bonds.

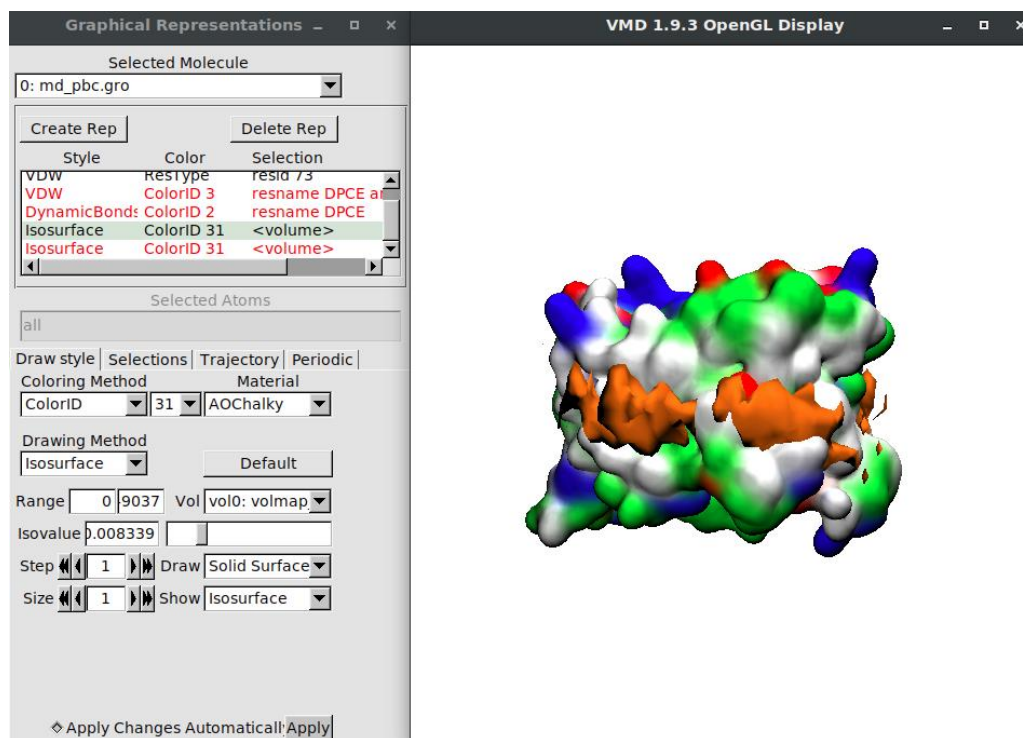
4.4. Occupancy maps

VMD can compute space occupancy maps over entire trajectories, and represent them as isosurfaces. In our case it makes sense to look at the occupancies of the entire ceramides and of the amide heads only. To compute a map open the Extensions -> Analysis -> VolMap Tool menu. Fill it in as the examples below and press 'Create Map' (once for each example).



You'll need to load the information in these files into your molecule. Select and right click your molecule name, in the main window, and choose 'Load Data into Molecule'. Browse to the `volmap_am.dx` and `volmap_dpce.dx` files and load them.

To visualize the occupancy maps go to Graphics -> Representations and add a new Representation. It won't matter what you put as the selection string, but in Drawing Method you must choose Isosurface. Set the remaining parameters as in the example below, and then play with the isovalue slider to see the effect of that threshold. Compare between AM beads or whole ceramide occupancies.



You should now be able to detect some conspicuous binding regions near the hydrophobic membrane core. Other things to try are to find frames where ceramides are bound to the spots identified in 4.4 and understand their mode of binding.

5. Conclusion

With this tutorial, you should have been able to use simple Martini tools to prepare a CG simulation.

The analyses were done with tools/methods not specific for CG. Still, given the large time scales achievable with CG, these kinds of analyses — which require extensive sampling for proper convergence — are well suited to CG trajectories.

Finally, a careful choice of molecular visualization can highlight the analysis outcome, sometimes even better than simpler plots.