Notes on how to use Lipowsky Lattice Membrane model

J. Hasnain

The purpose of this document is to provide a usage guide for the simulation of the interaction of two elastic membranes that are covered in proteins that interact with each other as the two vesicles are brought into contact.

1 Scientific Overview

The enclosed code follows the model of R. Lipowsky *et. al.*. Selected publications are enclosed in the "papers" folder, one of which details the implementation and theory and the other is a review of some of the results that have been obtained in the past. The basic idea of the algorithm is to perform a Monte-Carlo simulation of two elastic sheets and assign a probability of observing a protein to individual nodes on the membrane. By considering the probability of seeing a protein on a node, instead of simulating the particles explicitly, the code does not slow down at high protein coverages or when large numbers of proteins are present. In addition to simulating the "physical reality" of the membranes, each sheet can also be restrained in order to maintain a typical inter-membrane separation.

2 Requirements

This document assumes that you are running the program either on a Linux or Mac OS machine. These two operating system have access to bash, which allows for smooth compilation of the program and allows the user to install the Visual Molecular Dynamics program with which one can make movies of the simulation data.

3 Program Overview

At its core, after specifying all input parameters, the simulation produces 5 outputs:

1. **Screen output** about the progress of the simulation, this is for the users' edification.

2. **Data file** containing all energies, protein population numbers, membrane areas, and all other observables. This should be thought of as a "hard copy" of the screen output, along with considerably more entries.

3. **Trajectory file** containing the position of all nodes of the system. This trajectory file can be used for post processing data, and is helpful in seeing what the membranes are doing. To use this file, one ought to install Visual Molecular Dynamics (VMD) to view the membrane videos.

4. **Restart file** in which the program periodically stores the state of the system. As of now, it is not absolutely guaranteed that everything works, but in principle, it is possible to use this file as an input into the simulation, to start a run from where a previous simulation left off.

When the program is run, it needs to know where to store the aforementioned files. It is necessary to make the following folders:

- PATH/membranesims/
- PATH/membranesims/Trajectories/
- PATH/membranesims/Restarts/
- PATH/membranesims/DataFiles/
- PATH/membranesims/DataFiles/Hists/

The expression PATH can be any folder on your computer. For details on how to supply the directory "PATH/membranesims" to the program, see Section 5 on the description of the "--OutDir" option.

4 Compilation

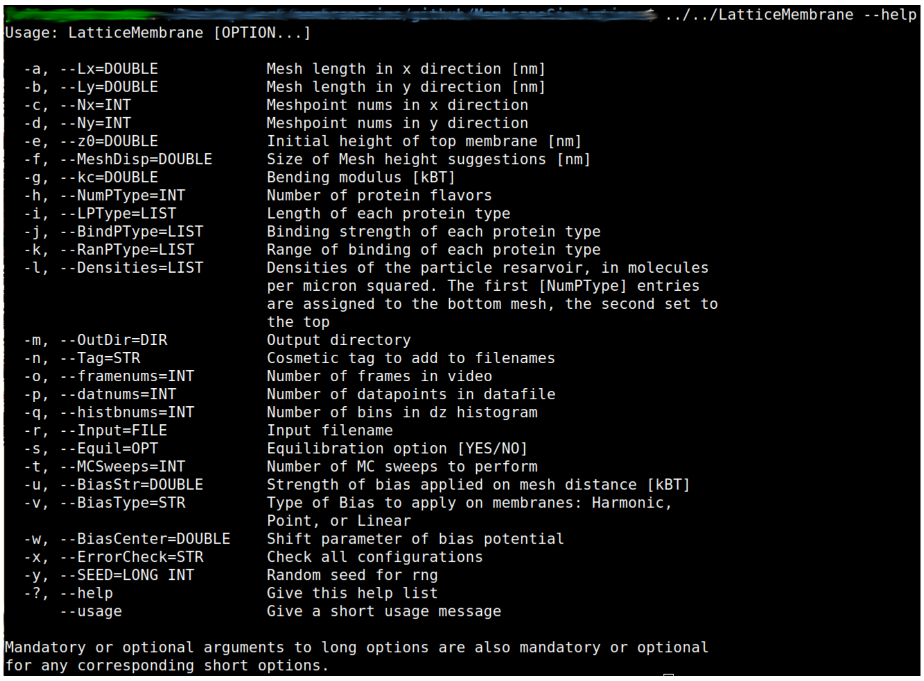
The simulation code is located in the "src" directory and consists of a number of text files with the ".c" or ".h" suffix. In addition to this, there is a one-line script called "compile.sh", which stitches the code together and makes an executable called "LatticeMembrane". The executable is the program. The program is going to be stored one level higher than the "src" directory. Whenever you make any changes to the program by editing the text in ".c" or ".h" files, the program will need to be recompiled in order for the changes to take effect.

5 Usage

5.1 Executing the program

The program is run by typing "./LatticeMembrane" into the terminal. Before running a simulation, try running "./LatticeMembrane --help". The result should be Fig. 1. When you run the program using the command "./LatticeMembrane" then the program will run using all of the default values. The default values are in the file called "parseparameters.c" in the function called *InterpretInputCommands* (line 127 to line 186).

There are three ways that the parameters of the simulation can be changed:



```
[redacted] ../../LatticeMembrane --help
Usage: LatticeMembrane [OPTION...]

  -a, --Lx=DOUBLE             Mesh length in x direction [nm]
  -b, --Ly=DOUBLE             Mesh length in y direction [nm]
  -c, --Nx=INT                Meshpoint nums in x direction
  -d, --Ny=INT                Meshpoint nums in y direction
  -e, --z0=DOUBLE             Initial height of top membrane [nm]
  -f, --MeshDisp=DOUBLE       Size of Mesh height suggestions [nm]
  -g, --kc=DOUBLE             Bending modulus [kBT]
  -h, --NumPType=INT          Number of protein flavors
  -i, --LPType=LIST           Length of each protein type
  -j, --BindPType=LIST        Binding strength of each protein type
  -k, --RanPType=LIST         Range of binding of each protein type
  -l, --Densities=LIST        Densities of the particle resarvoir, in molecules
                              per micron squared. The first [NumPType] entries
                              are assigned to the bottom mesh, the second set to
                              the top
  -m, --OutDir=DIR            Output directory
  -n, --Tag=STR               Cosmetic tag to add to filenames
  -o, --framenums=INT         Number of frames in video
  -p, --datnums=INT           Number of datapoints in datafile
  -q, --histbnums=INT         Number of bins in dz histogram
  -r, --Input=FILE            Input filename
  -s, --Equil=OPT             Equilibration option [YES/NO]
  -t, --MCSweeps=INT          Number of MC sweeps to perform
  -u, --BiasStr=DOUBLE        Strength of bias applied on mesh distance [kBT]
  -v, --BiasType=STR          Type of Bias to apply on membranes: Harmonic,
                              Point, or Linear
  -w, --BiasCenter=DOUBLE     Shift parameter of bias potential
  -x, --ErrorCheck=STR        Check all configurations
  -y, --SEED=LONG INT         Random seed for rng
  -?, --help                  Give this help list
      --usage                 Give a short usage message

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.
```

Figure 1: Help screen. This is a list of all variable parameters in the simulation.

1. Directly edit *InterpretInputCommands* in the file "parseparameters.c", and then recompile the program.

2. Type "./LatticeMembrane --Param_Opt=VALUE". Note that you need to use two "-" characters.

3. Perform "./LatticeMembrane --Input=FILE", which reads a restart file generated from a previous simulation and continues the simulation.

The list above is ordered from lowest priority to highest. This means that if you use an input file, then ALL parameters in the input file are used, regardless of any other settings. If you use command line arguments (item 2), then default values are overwritten by the "VALUE" that you supplied. Finally, if you do not specify a parameter, then the default value is used.

It is strongly recommended, that after it is decided where output should be written (see Sec. 3), that the default value for "OutDir" is set to that location (line 174 in parseparameters.c).

5.2 Parameter description

In the following, each option is briefly explained and recommendations are made for their values. The options in the program are grouped by: membrane parameters, protein parameters, output options, and Monte-Carlo parameters. All lengths in the simulation are in nanometers and energies are expressed $k_{\mathrm{B}}T$, except for the particle density parameters, which are in molecules/μm^2. As mentioned earlier, all parameters can be changed using the command line (when the program is executed), in the parseparameters.c file (which requires you to recompile to program), or through the use of an input file.

5.2.1 membrane parameters

- --Lx, --Ly:
  Sets the length of the membrane in the x and y direction, respectively, in nanometers. It is **not** recommended to use different lengths. For conceptual reasons, the program will deliver an error if L_x does not equal L_y. Example usage: "./LatticeMembrane --Lx=300 --Ly=300" creates a pair of $300 \times 300\,nm^2$ membrane patches.

- --Nx, --Ny: Discretization of the membrane. "--Nx" determines the number of discretization points used to discretize the x direction of the membrane. It is **strongly** recommended that the number of nodes are chosen so that the membrane is discretized into square patches that are $4 \times 4nm^2$ in size. Using the example above, $300nm$ should be chopped-up into $N_x = N_y = 75$ segments. In terms of simulation time, the program only cares about the number of nodes. Small, medium, and large simulations roughly correspond to $N_x \times N_y = 1600$, 3600, 10000 nodes, respectively.

- --z0: Initial height difference between the two membrane patches. Initially, the two membrane patches are separated a distance z_0 from each other. The bottom mesh starts at $-z_0/2$ and the top mesh is at $z_0/2$. Both meshes are initially perfectly flat.

- --MeshDisp: Size of typical mesh-node-displacement (in nm). For each Monte-Carlo move, the program will suggest a new position for a randomly chosen node. The MeshDisp option determines how big the typical displacement will be. The default setting is 0.4 and should only be changed if, after running the code, you find that Monte-Carlo acceptance rate is close to 0% or close to 100%. An acceptance rate between $50\% - 70\%$ is considered to be good.

- --kc: Bending modulus of the membranes. A bending modulus value of $20\,k_\mathrm{B}T$ is considered to be physiological conditions.

For example, the command "./LatticeMembrane --Nx=40 --Ny=40 --Lx=160 --Ly=160 --kc=30 --MeshDisp=0.1" overwrites the default mesh parameters and creates pair $0.0256\mu m^2$ membrane patches with a bending modulus of $30\,k_\mathrm{B}T$, consisting of 1600 nodes, and performs Monte-Carlo moves using a displacement of $0.1nm$.

5.2.2 particle parameters

- --NumPtypes: stands for Number of Particle Types. This sets the number of particle *species* in the system. A value of 0 means that the vesicles/membranes have no membrane proteins on them. In the simulation, the there are proteins in the "Bottom" membrane and on the "Top". When "NumPtypes=1", then there is 1 protein species on the bottom membrane and another at the top. Most of the remaining particle parameters will depend significantly on the number of particle types.

- --LPtypes: stands for Length of Particle Types. This sets the length of each of the protein species on each of the membranes and therefore has $2\times$ [NumPtypes] entries. The format for this and the following parameters is of type <u>LIST</u>,

  – The command "./LatticeMembrane --NumPtypes=1 --LPTypes='10.0 3.5' " defines a system in which each membrane has one particle species. The particles on the bottom membrane have length $10nm$ and the proteins on the top membrane have length $3.5nm$. Note the single quotes around the argument supplied to "LPTypes". 1 length needs to be specified for each particle species for each membrane. For a system defined by the command "./LatticeMembrane --NumPtypes=2 --LPTypes='10.0 3.5 4.5 100.0' " species 1 on the *bottom* mesh has length $10nm$, species 2 on the *bottom* mesh has length $3.5nm$, species 1 on the *top* has length $4.5nm$ and species 2 on the *top* mesh has length $100nm$. Unless otherwise specified, all parameters of type LIST must be given in the form of a string, *i.e.* using quotes, and then the first "NumPtypes" entries correspond to the proteins in the bottom mesh and the second set of "NumPtypes" entries are for the top mesh.

- --BindPType: stands for Binding energy of Particle Type. This option determines how particles of the same species interact. The parameter is of the type <u>LIST</u>. In this case the number of entries in the quotation marks is NumPtypes. If a protein of species 1 on the top mesh is near to a protein of species 1 on the bottom mesh, then they can form a bond with an energy specified by this option. The command "./LatticeMembrane --NumPtypes=1 --LPTypes='10.0 10.0' --BindPType='4.0' " defines one binding protein species, each of length $10nm$ that bind to each other with energy $4\,k_{\mathrm{B}}T$. The command "./LatticeMembrane --NumPtypes=2 --LPTypes='10.0 2.0 15.0 12.0' --BindPType='0.0 4.0' " creates 2 protein species. The first species is non-binding, and consists of $10nm$ proteins on the bottom mesh and $15nm$ proteins on the top mesh. The second species has a binding energy of $4\,k_{\mathrm{B}}T$ consists of a short $2nm$ linker and and long $12nm$ linker.

- --RanPType: stands for Range Particle Type. This option determines the interaction range of binding proteins. The parameter is of the type <u>LIST</u>. In this case the number of entries in the quotation marks is NumPtypes. If the corresponding binding energy is not 0, then this parameter defines the interaction range of the bonds. The typical value for the interaction range is $4nm$, which is roughly the diameter of the proteins. One should always specify the binding ranges if "BindPType" is also defined. The command "./LatticeMembrane --NumPtypes=1 --LPTypes='10.0 10.0' --BindPType='4.0' –RanPType='4.0' " defines the binding proteins from the example above and creates bonds only if the proteins are within $4nm$ of each other. The command "./LatticeMembrane --NumPtypes=2 --LPTypes='10.0 2.0 15.0 12.0' --BindPType='16.0 4.0' --RanPType='1.0 6.0' " creates 2 binding protein species. The first binding pair forms bonds of strength $16\,k_{\mathrm{B}}T$ if the ends of the proteins are within $1nm$ of each other.

The second pair of binders form bonds of strength $4\,k_{\mathrm{B}}T$ but can be as far away as $6.0nm$ from each other.

- --Densities: defines the number of molecules per micron squared of each particle species on the bottom and top membrane. The argument is of type <u>LIST</u> and has 2× [NumPtypes] entries. The first [NumPtypes] entries correspond to densities of the molecules on the bottom mesh, the remaining [NumPtypes] entries define the densities of the top mesh. The command "./LatticeMembrane --NumPtypes=2 --Densities='100.0 1000.0 10.0 10000.0' " creates 2 protein species, with the bottom vesicle having a molecule density of species 1 $100\mu m^{-2}$ and a density $1000\mu m^{-2}$ of species 2. The top membrane has a low density of $10\mu m^{-2}$ for species 1 and a high density of $10000\mu m^{-2}$ of the second species. From discussion and literature review, it is typical to observe a density $1000\mu m^{-2}$ on a GUV, whereas a cell can have as high a density as $10000\mu m^{-2}$.

5.2.3 Output options

- --OutDir: Tells the program where to store the output files. The directory needs to have the listed folders and subfolders as outlined in Section 3. It is recommended that the directory location is entered into parseparameters.c. The directory can be changed using "./LatticeMembrane --OutDir=' /MyProjects/MembraneDat/'.
- --Tag: Helpful option to keep track of the different types of simulations that are conducted. The command "./LatticeMembrane --Tag='OneBinder_OneNonBinder' ", adds the string "OneBinder_OneNonBinder" to all output files to help keep track of what type of simulation was conducted, and to make data processing easier.
- --framenums: Determines how many configurations should be saved over the course of the simulation ([OutDir]/Trajectories/). It is suggested to store at most 2000 frames. More than this costs a great deal of memory.
- --datnums: Sets the number of times data should be output to the generated data file ([OutDir]/DataFiles/). This data is used to compute averages and free energies, for example. Storing 20000 data points per simulation is reasonable.
- --histbnums: Sets the number of bins to histogram height fluctuations ([OutDir]/DataFiles/Hists/). There is some code that computes the height fluctuations of the membranes and gathers statistics on this. A value of 0 switches these routines off. The code for this has not been fully tested, but can be made to work if one wants to compute histograms.

5.2.4 Monte Carlo options

- --Input: the command "./LatticeMembrane --Input=restart.rest" will look for the file "restart.rest" in the current directory, and continue simulating from the moment the simulation that created the restart file stopped. The files generated periodically in "OutDir/Restarts/" can be used as input files. The restart file routines have not been updated recently so there is no guarantee that they will work.

- --Equil: does nothing.
- --MCSweeps: determines how long the simulation will run. If the number of nodes used in the simulation N_{Tot}, then 1 Monte Carlo sweep consists of N_{Tot} Monte-Carlo moves, *i.e.* after each sweep, every node will have tried to move once, on average. A simulation consisting of 10^3 MC Sweeps is short, 5×10^4 sweeps is a medium run, and 10^6 is a long run.
- --BiasStr: most membranes pairs, if left on their own, will repel each other and float infinitely far away from each other. In order to avoid this, the one can pin each membrane a fixed distance away from each other. The BiasStr parameter sets how strongly the membranes are held in place. This parameter is the spring constant of a harmonic potential that holds the membrane in place. A value of Biasstr $= 10\, k_{\mathrm{B}}T$ is usually used.
- --BiasType: There are many different ways to hold the membranes in place. Currently, the (default) harmonic bias is guaranteed to be correctly implemented, and gives the best results. If other types are to be used, then one should carefully update the code.
- --BiasCenter: inter-membrane distance that the bias potential tries to impose. Two membranes will always have a preferred separation (which is sometimes infinitely far away). By specifying a BiasStr and a BiasCenter, one can learn about intermediate states and study how the membranes reach stable states. The command "./LatticeMembrane --BiasCenter=15 --BiasStr=5", the code attempts to keep the membranes at a distance of $15nm$ with a spring constant of $5\, k_{\mathrm{B}}T$.
- --ErrorCheck: A number of coding errors can be easily found by double checking certain calculations. By changing the default value of ErrorCheck from "no" to "YES", the program tries to catch obvious mistakes in the code. This comes at the price of making the simulation very slow. The option "Yes" (note capitalization), the program only checks once every Monte Carlo sweep. The option "YES" makes a check every sweep. After every change in the code, it is strongly recommended to run 100 sweeps using this option, since it might save a lot of debugging time.
- --SEED: Monte Carlo simulations rely on random number generators. In order to make sure that each simulation is reasonably random, it is necessary to supply the program with an actual random number. The command "./LatticeMembrane --SEED=${RANDOM}" gives the program a random seed to initialize the random number generator with. This is absolutely necessary for production runs in order to get clean data. The default behavior of the program is to use the same random seed every time the program is executed. This is helpful from a coding and testing perspective since the results of the simulation will be identical if no other parameter changes.

6 Data Analysis

Ongoing