MLLPA: A Machine Learning-assisted Python module to study phase-specific events in lipid membranes

Vivien Walter, Céline Ruscher, Olivier Benzerara, Fabrice Thalmann

MLLPA: A Machine Learning-assisted Python module to study phase-specific events in lipid membranes

Vivien Walter[*], Céline Ruscher [†], Olivier Benzerara[†],Fabrice Thalmann[†]

February 23, 2021

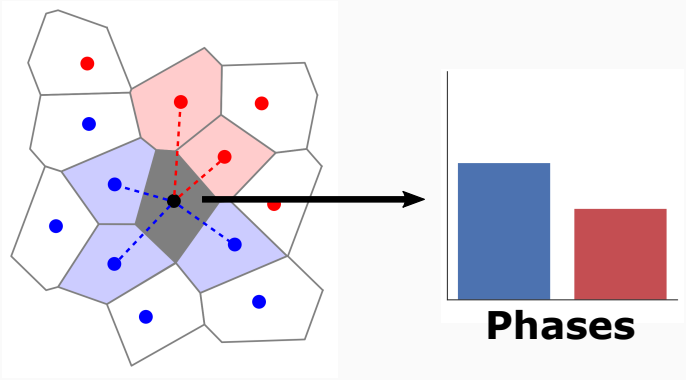
Abstract

MLLPA is a new Python 3 module developed to analyse phase domains in a lipid membrane based on lipid molecular states. Reading standard simulation coordinate and trajectory files, the software first analyse the phase composition of the lipid membrane by using Machine Learning tools to label each individual molecules with respect to their state, and then decompose the simulation box using Voronoi tessellations to analyse the local environment of all the molecules of interest. MLLPA is versatile as it can read from multiple format (*e.g.* GROMACS, LAMMPS) and from either all-atom (*e.g.* CHARMM36) or coarse-grain models (*e.g.* Martini). It can also analyse multiple geometries of membranes (*e.g.* bilayers, vesicles). Finally, the software allows for training with more than two phases, allowing for multiple phase coexistence analysis.

Keywords: Lipid membrane analysis, Molecular dynamics, Phase transition, Tessellation, Machine Learning.

[*]Department of Chemistry, King's College London, Britannia House, 7 Trinity Street, London, UK

[†]Institut Charles Sadron, CNRS and University of Strasbourg, 23 rue du Loess, F-67034 Strasbourg, France



A new python software uses machine learning to characterise efficiently the individual thermodynamic states of lipids in membranes. It offers new routes for microscopic exploration of lipids mixtures and for the investigation of the interaction between lipids and external molecules or macromolecules.

INTRODUCTION

Lipid membranes play a key role in the existence and organisation of the cells, and their study is critical to fully understand the mechanisms of life[1,2]. As lyotropic liquid crystals, phospholipid molecules composing about 50% of the membrane weight display fascinating thermodynamic properties, notably phase transitions[1,3–5]. These transitions, either driven by temperature or molecular composition, separate the system free-energy landscape into different phases where lipid molecules adopt a subset of typical conformations, depending on the amount of order inside the membrane (*cf.* Fig. 1)[6–9].

In all but the simplest situations lipid membranes display heterogeneities, related to thermodynamic coexistence, dynamic domain coarsening or non-lipid inclusions. Assigning individual lipids to a state, or a phase, is therefore highly desirable. However, the usual tools used in molecular dynamics (MD) simulations, such as the measurement of the area per lipid or of the tail order parameter, are limited in that they can only determine the average phase of the whole membrane or lipid group[10,11]. MLLPA, short for Machine Learning-assisted Lipid Phase Analysis, is a free, open source module for Python 3 that has been developed to extract the membrane molecules atomic positions from standard simulation files, and analyse their conformation via Machine Learning (ML) with the purpose of determining the most likely thermodynamic phase these molecules belong, at every single frame of the simulation. Once the lipids have been properly identified and given a state or phase label, MLLPA is then able to analyse the average environment of each molecule in the membrane using Voronoi tessellations.

The interest for ML-based analysis tool is growing exponentially in all science fields, as it has been widely proven to be a versatile and robust tool that drastically reduce the bias brought by the user in the selection of the parameters to consider[12,13]. ML has been already applied to lipid molecules and membranes to predict nanostructures or predict and locate thermodynamic phases interfaces[14–17]. Other non-ML-based approaches to detect phase transition are still being developed[18], but they remain scarce. However, all these methods either lack specific phase labelling or lack capacities to identify the state of individual lipid molecules in the membrane. To our knowledge, the first attempt at identifying via ML the

nature of the phase by analysing the configuration of the lipids was published by some of us[19].

Tessellations are powerful and useful representations of the geometrical organisation of complex biological systems, such as tissues[20]. MLLPA uses the Voronoi tiling[21,22] to define unambiguously the neighbour list of each lipid molecule in the membrane. The applications of tessellations to analyze MD simulations of lipid membranes are also relatively common, especially to calculate the area per lipid[23,24].

In order to operate, the different functions of MLLPA rely on two main scientific libraries. First (1) the library *scikit-learn* for Python[25] implements the different ML algorithms required to analyse the lipid molecular conformations: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Gaussian Naive-Bayes (NB) and Classification and Regression Tree (CART) algorithns, and second (2) the library *voro++* for C++ that we use through the Python wrapper Tess[26,27] performs the Voronoi tessellations of the membrane and maps the neighbours of each membrane molecule. To optimise its versatility in working with different MD simulations softwares, MLLPA is based on the library *MDAnalysis* for Python[28,29] to open the coordinate and trajectory files, and extract the atom positions and properties (*i.e.* ID, names, types and masses). While the modules for reading simulations files (*MDAnalysis*), performing a Machine Learning analysis (*scikit-learn*) and tessellating the simulation box (*voro++*) are already freely available, none of them were designed to specifically analyse lipid membrane, and coming from very different application domains were not designed to be used together as well. MLLPA aims at bridging the Molecular Dynamics and Machine Learning toolkits by offering a smooth and ergonomic user interface. As shown in this paper, the preparation steps and computations required for these analysis are far from trivial, making MLLPA a superior and essential tool for lipid phase transition applications in molecular dynamics simulations. MLLPA has been specifically programmed to handle all types of lipid membranes and molecules interacting with it, and provides a fast and reliable analysis of the phase composition and the interactions with phase domains. MLLPA is free to use and modify under the terms of the GNU General Public License (version 3.0), and the whole code is available on GitHub[30]. MLLPA is easy to install, by automatically handling and installing all the required packages. A speed test ran in the simulation section also demonstrates that

analysis with MLLPA are fast even on a non-parallel desktop computers.

In the next section, we describe the algorithms and architecture of the MLLPA software. Cases studies using MLLPA are then presented in the second section, Validation Simulations. The lipids that were considered in the present work are 1,2-dipalmitoylphosphatidylcholine (DPPC), 1,2-dioleoylphosphatidylcholine (DOPC) and cholesterol. Pure DPPC bilayers display a main gel/fluid melting transition at 41°C while DOPC bilayers remain in a fluid state down to *ca* -20°C. Cholesterol alone does not form bilayers but interact very specifically with saturated lipids such as DPPC. The organisation of phospholipid-cholesterol mixtures is specific and very relevant in the field of membrane biophysics. DOPC-DPPC-cholesterol mixtures are also known to display a rich phase domain behaviour depending on temperature and composition. We also briefly considered 1,2-DimyristoylPhosphatidylEthanolamine (DMPE) which melts at 50°C and 1,2-dipalmitoylphosphatidylethanolamine (DPPE) which melts at 74°C. In the case of pure phospholipid the main melting transition often separates a low temperature gel phase (L_β or tilted $L_{\beta'}$) from a fluid phase L_α. In the context of cholesterol-phospholipid mixture, one distinguishes two "fluid phases": a disordered phase L_d very similar in structure and order to the fluid phase L_α, and a specific liquid ordered phase L_o similar to the gel phase in terms of chain order, but to the fluid phase in terms of fluidity. Readers unfamiliar with lipid molecules and lipid biophysics can refer *e.g.* to references[1,2,4,5] for an introduction.

DESCRIPTION

When used directly with MD simulation softwares outputs, the program requires for each simulation analysed at least two files as input: (i) a coordinate file, and (ii) a structure file. The program can also analyse full trajectories, in which case a (iii) trajectory file is required. For instance, in the GROMACS format, these files correspond respectively to a *.gro (or *.pdb), a *.tpr and a *.trr (or *.xtc) files. It is also possible to directly load position arrays in MLLPA. For the machine learning prediction of the states of the lipids, MLLPA does not require any specific orientation of the membrane. For the neighbour analysis in a lipid bilayer, this one should be oriented normal to the z-axis. It is essential to note here that all

periodic boundary conditions (PBC) abnormalities, *e.g.* molecules cut in several groups of atoms through the simulation box, should be corrected before being processed by MLLPA. Besides that, MLLPA supports natively the presence of PBC for mapping the neighbours.

To analyse a system, the typical sequence of processes to call in MLLPA is as follow: (1) train the Machine Learning model, by (1a) extracting the molecules representing each phase from the simulation file and then (1b) training the models on them; (2) analyse the states of the molecules in the simulation, by (2a) extracting the molecules from the file of unknown composition and then (2b) using the previously trained models to predict the phases. Once these steps are done, the systems are ready and can (3) be tessellated to (3a) extract the list of neighbours for every molecule and (3b) analyse the phase composition of these neighbours. Finally, (4) the results can be saved in files.

A user's guide providing tutorials and clear descriptions of all the functions and classes found in MLLPA is distributed with the program. The tutorials and the descriptions of the functions and classes are available on the GitHub of the project and its website[30]. Some simulation files of a 256 DPPC bilayer, used in the Validation Simulations part of this document to train the models, have been uploaded on Zenodo in order to test MLLPA[31]. In this section, the algorithms of each functionality of the program is discussed thoroughly.

Generation of the systems

The collection of positions, information and eventually frames in the case of a trajectory extracted from simulation files is referred as a **system** in MLLPA. The function used to process simulation files is named *openSystem* and takes as argument the coordinate and structure files path. It is essential to note that, in order to increase the processing speed and to conform with the requirements of Machine Learning inputs, a system in MLLPA is specific to only one type of residue/lipid. The name of the residue to extract is therefore a mandatory input of the *openSystem* function. Another required input is the neighbour rank for ranked intra-molecular distances calculations (see below). A flowchart of the processes involved in the *openSystem* function can be found in the Supplementary Materials.

Processing the simulation files

The function *openSystem* calls *MDAnalysis* to open the simulation files and extract the following properties:

- The 3-D positions of all the atoms of the given molecule type. If a trajectory file is provided, the positions are read for each selected frame.
- The list of all atomic bonds in the given molecule type.
- The mass of every atom in the given molecule type.
- The name and ID of the molecules and their atoms for proper identification.

Since the file input functions of MLLPA are based on the library *MDAnalysis*, MLLPA shares with this library its list of compatible file formats, and the formats handled by the function *openSystem* are restricted to this list. The exhaustive list of file formats compatible with MDAnalysis can be found in its official documentation[32]. Regardless of the format, the coordinate and trajectory files should include the positions of all atoms on all frames, and the structure file should include the list of bonds, masses, names and IDs of all atoms. In the case where the coordinate file would also include the information extracted from the structure file, the coordinate file can be re-used.

To maximize its versatility and compatibility, MLLPA is capable to directly work with position arrays. This can be useful when the simulation file format to be analysed cannot be opened by MDAnalysis. For this purpose, the function *systemFromPositions* has been implemented. Other sub-functions have been provided to help the users formatting all inputs for MLLPA. More information on these functions can be found in the online documentation of MLLPA[30].

The positions are stored in an array of dimension *(number_frames, number_molecules, number_atoms, number_space_dimensions)*, with 3 space dimensions in the usual Cartesian system of coordinates. In the case where no trajectory file has been provided, the number of frames is set to 1. Because of the tight correlation between the hydrogen atoms and the heavy atoms they are bonded with, hydrogen atoms are removed from the positions array in the case of all-atom simulations. For unified atoms and coarse-grained simulations, all

particles are kept in the array. Atomic bonds, masses, names and IDs are stored as arrays in a dictionary.

Creating the configuration spaces

The next step prepares the positions for the Machine Learning analysis, either for training or prediction purposes. Analysing the phase of lipid membranes via Machine Learning requires two sets of input parameters, which are called configuration spaces: the positions of the atoms of the molecules in a cylindrical system of coordinates, called **coordinates** for short, and the intra-molecular distances set for a specific pair rank, called **distances** for short. These two feature or configuration spaces were introduced in detail in our previous paper[19].

Once all the molecules conformations have been processed and mapped to their configuration spaces, the *openSystem* function returns as an output an instance of a class named *System*. The instance of the *System* class contains the positions, coordinates and distances of the molecules, as well as a dictionary summarizing all the information extracted from the structure file.

Cylindrical Coordinates The positions of the atoms of all molecules in cylindrical coordinates are computed from the positions array, provided in Cartesian coordinates, and from the atomic masses array stored in the dictionary extracted from the structure file.

In order to properly decorrelate the state of the molecule from its position and orientation in the membrane or the simulation box, the first step is to compute and subtract the position of the center of mass of the molecule from the positions of the atoms. This is done molecule by molecule and frame by frame using operations on matrices (*i.e* arrays). Any orientation bias of the molecule in the simulation box is also removed during the second step. To do so, the molecule is re-orientated to with its longer axis of inertia parallel to the z-axis. This operation is done by computing the gyration tensor $\mathbf{G}$ of the molecule

$$\mathbf{G} = \frac{1}{\sum_i^N m_i} \begin{bmatrix} G_{XX} & G_{XY} & G_{XZ} \\ G_{YX} & G_{YY} & G_{YZ} \\ G_{ZX} & G_{ZY} & G_{ZZ} \end{bmatrix}, \tag{1}$$

with $\sum_i^N m_i$ the total mass of the molecule, and G_{RS} the elements of the tensor defined as the mass weighted sum of the quadratic product of the atoms coordinates along the axis R and S

$$G_{RS} = \sum_i^N m_i R_i S_i. \tag{2}$$

The eigenvectors $\mathbf{e}_a$, $\mathbf{e}_b$, $\mathbf{e}_c$ and the eigenvalues λ_a, λ_b, λ_c of the gyration tensor are then computed from the matrix $\mathbf{G}$

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{a,X} & \mathbf{e}_{a,Y} & \mathbf{e}_{a,Z} \\ \mathbf{e}_{b,X} & \mathbf{e}_{b,Y} & \mathbf{e}_{b,Z} \\ \mathbf{e}_{c,X} & \mathbf{e}_{c,Y} & \mathbf{e}_{c,Z} \end{bmatrix}, \tag{3}$$

where the indices a, b and c are chosen to satisfy $\lambda_a < \lambda_b < \lambda_c$. Finally, the (orthogonal) matrix $\mathbf{E}$ is inverted and multiplied to the position vectors of all atoms $\mathbf{P}$, properly rotating the molecule so its longest axis is parallel to the z-axis:

$$\mathbf{P}' = \mathbf{E}^{-1}\mathbf{P}. \tag{4}$$

All the linear computations done here are based on the scientific library *NumPy*[33].

In the final step, the 3-D Cartesian positions of the rotated molecules are converted to cylindrical coordinated. As demonstrated in our previous paper[19], the azimuth in the cartesian coordinates of the molecule is not necessary for Machine Learning analysis, and is therefore removed. The result is an array of dimension *(number_frames, number_molecules, number_atoms, 2)*.

Ranked Intra-molecular Distances The intra-molecular distances are computed from the positions array provided in Cartesian coordinates, and from the atomic bond list stored in the dictionary extracted from the structure file.

For every atom of the molecule, distances are calculated between the atom considered and each atom forming a pair with it at a given rank along the molecular chain. A pair of rank R is defined as two atoms separated by $R-1$ atoms along the chain, as illustrated in

Fig. 2. For example, two atoms sharing a direct covalent bond form a pair of rank 1, while two atoms connected to the same 3rd atom form a pair of rank 2.

The first step in the generation of the distances is to build the list or map of all pairs inside the molecule with the same given rank. The algorithm used to generate this map is illustrated in the Supplementary Materials. This has to be done only once per lipid type when the instance *system* is created. In the case of ring molecules, only the smallest rank between two atoms is considered.

Once the map is available, distances are calculated one pair at a time, using the coordinate space Euclidian distance. The result consists in an array of dimension *(number_frames, number_molecules, number_distances)*.

Machine Learning training and predictions

The core requirement to recognize phase domains in a lipid membrane is the capacity to determine the most likely phase domain a given lipid belongs to, based on its conformation state. MLLPA performs this analysis by resorting to Machine Learning algorithms. These algorithms are all made available by the scientific library *scikit-learn*. A total of four models are used in MLLPA: the K-Nearest Neighbors algorithm (KNN) trained on coordinates, Naive Bayes Gaussian algorithm (NB) trained on distances, the Support Machine Vector algorithm (SVM) trained on both. Finally a Classification and Regression Trees (CART) algorithm is used to compare the predictions of the four other models, to finally decide the predicted state. The reason why a two-steps ML analysis of the data-set is required was discussed in length in our previous work[19]. Briefly, some of the ML algorithms have a great efficiency for predicting one of the phase but not the other. By using 4 models and by combining their decision with a classification tree, MLLPA optimises its accuracy for all phases.

MLLPA is not provided with any built-in ready to use model. Users are required to train their models on their own systems first. This allows for a better accuracy of the Machine Learning predictions and makes it possible to use the tool without restriction regarding the phases, or the lipid molecules architecture and implementation.

Using systems to train a model

In order to train the models implemented in MLLPA, one must provide exactly 1 system (as generated above) for each phase to analyse and train the models on. Minimum number of phases (or states) is 2, but there is in principle no upper limit in the number of phases that can be trained on. The function used to train the Machine Learning algorithm is called *generateModel*. It takes as arguments a list of all the instances of the *System* class to analyse, as well as a list of the phases that these systems are meant to represent.

In the initial step, the input coordinates and distances arrays are reshaped into arrays of respective dimensions *(number_frames* $\times$ *number_molecules, number_atoms* $\times$ *number_cylindrical_coordinates)* and *(number_frames* $\times$ *number_molecules, number_distances)*. This process is repeated for each of the k input systems, and all the arrays are then stacked into two massive arrays of dimensions $(\sum_i^k$ *number_frames*$_i$ $\times$ *number_molecules*$_i$*, 2*$\times$ *number_atoms)* and $(\sum_i^k$ *number_frames*$_i$ $\times$ *number_molecules*$_i$*, number distances)*. The name of the phases provided in the input are used to generate the array of classes of dimension $(\sum_i^k$ *number_frames*$_i$ $\times$ *number_molecules*$_i$*,)*. These three arrays constitutes the input set of the Machine Learning training process. A flowchart describing the generation of model is provided in the Supplementary Materials.

Prior training, the input set is randomly shuffled before being split into three subsets: a subset for the main training on the four models (KNN, NB, SVM distances and SVM coordinates), a subset to train the final prediction proposed by the CART algorithm, and the subset for verification and scoring. The respective sizes of these subsets are 60, 20 and 20% of the input set[1].

Once the main training has been completed, the four ML models are asked to predict the states of the intermediate training subset conformations. The four prediction arrays are then used to form a unique array on which the CART algorithm is trained. After this step, MLLPA can be considered as properly trained and ready to predict the phase or state of the related molecule type. The accuracy is finally assessed by using the third remaining subset. States are predicted using the two steps prediction routine, and compared to the known

[1]The proportional sizes of the subsets given here are the default values in MLLPA. Users can edit the values in the argument of the function *generateModel*.

labels. The accuracy is calculated as the ratio of successful predictions.

In order to obtain trustful scores and models, all the subsets are shuffled and split 10 times and the final score is given as the average of all the 10 scores obtained. The model returned is the model which achieved the best score during the repeated attempts. It is critical to note here that the accuracy scoring performed by MLLPA depends entirely on the quality of the training set, and on the accuracy of the state labelling. While ML algorithms are robust enough to handle a limited amount of incorrectly labelled objects in the training subsets, a too large number of incorrect label leads to incorrect predictions. It is therefore necessary to check the scores of the trained model prior to using it.

The *generateModel* function produces two types of output: first, it generates a dictionary containing the different Machine Learning models that can be directly used to predict the phases of unknown system (see section below). It can also generates a model file to store the trained model for later use. The model file, saved under the format *.lpm (for lipid phase model), is a simple archive file that can be opened like a *.zip file, or using the *readModelFile* function implemented of MLLPA. The content of this file are: a (i) coordinates, a (ii) distances and a (iii) states list which include the whole input set used to train the model, all saved as *.csv files; as well as a metadata *.xml file that includes the scores of all parts of the training and the different relevant information to train the model again on the content of the *.csv files. The models are not saved as binary files (*e.g.* Pickle or JSON outputs) to prevent any compatibility issues with other versions of *scikit-learn*[2] or any other scientific library required in MLLPA.

Prediction of the phases of a system

Once a model has been trained with MLLPA on a molecule type, it can be used to predict the state of molecules of the same type in unknown systems. This is done by calling the function *getPhases*. The function reads the instance of the *System* class provided, as well as either the dictionary of trained models or the path to the model file to use. If a model file (*.lpm) is provided, MLLPA extracts all the training sets and parameters and re-train the

[2]To optimise the compatibility and reproducibility, the versions of MLLPA and scikit-learn are saved in the metadata file within the model file.

models on them.

Similarly to the model training function, the input coordinates and distances arrays of the system to analyse are first reshaped. The arrays are then analysed by the trained models and the predictions of each of the models are processed by the trained CART model to generate the final prediction on the state of the molecules.

Most biomolecules as well as some lipids do not undergo any phase transition in the temperature range studied. For example, in a DPPC/DOPC mixture simulated from 290 to 360 K, the DOPC shall remain the whole time in the fluid phase since its melting temperature is 256 K. As a consequence, the Machine Learning models implemented in MLLPA cannot be trained on DOPC without further efforts that one can perceive as unnecessary. To circumvent these specific cases and still allows for later mapping of the membrane, the function *setPhases* can also be found in MLLPA. Instead of predicting the state of the molecules, MLLPA enforces the state provided as an input. The output of both *getPhases* and *setPhases* functions are the updated instances of the *System* class.

Voronoi tessellation for neighbours identification and analysis

The final part of the MLLPA program analyses the position of the molecules in the simulation box and builds a Voronoi tessellation of the space to connect the molecules to their direct neighbours. The tessellation of the space is performed using the centres of mass of the molecules of the system. Using the predictions of Machine Learning, the phase composition of the neighbours is directly obtained.

Ghost lipids and tessellations

The tessellation of the membrane is performed by the *doVoro* function. The function takes as an input the list of all instances of the *System* class to analyse the geometry of the membrane and extract the centres of mass of the molecules to process. Using the centres of mass instead of specific atoms at the hydrophilic/hydrophobic interface of the membrane allows MLLPA to work with all types of molecule discretisation, from all-atom to coarse-grain molecules, and to include hydrophobic molecules inserted inside the bilayers.

The 3-D tessellation of a lipid bilayer must circumvent a number of difficulties, due to the asymmetric geometry and the presence of water layers. Indeed, lipids from the upper leaflet have close neighbours underneath but no neighbour above, and likewise for the lower leaflet. Because of the periodic boundary conditions (PBC) of the simulation box, lipids from the other side of the simulation box end up incorrectly identified as neighbours by the tessellation. This is particularly true in the case of a vesicle, as shown in Fig. 3 (a, b).

MLLPA is designed to avoid this situation. This is done by generating "ghosts" for all the molecules in the system. Ghosts are mirrors of the molecules with respect to the membrane interface with the solvent, which are temporarily included in the Voronoi tessellation. Due to their close proximity to the original lipids, ghosts get included into the neighbours list, substituting for the unwanted remote PBC images (*cf* Fig. 3 (c, d)).

Ghosts are generated using the centers of mass of all the molecules in the membrane. The vector between the center of mass of the individual molecules $\boldsymbol{r}_g$ and the center of mass of the total membrane $\boldsymbol{r}_G$ is then calculated. In the case of a vesicle, the center of mass of the total membrane reduces to a single point which is computed (Fig. 3 (e)), while in the case of a bilayer the "center of mass" is a Z-plane separating the two leaflets (Fig. 3 (f)). As a consequence, the vectors will always be taken parallel to the z-axis (normal to the bilayer). It is therefore essential to specify in the input which type of geometry should be analysed. After this step, the leaflet in which each lipid is located is determined.

Once the vectors have been computed, the position $\boldsymbol{r}_{\max}$ of the furthest atom of the molecule from the center of mass of the bilayer along the unitary vector $(\boldsymbol{r}_g - \boldsymbol{r}_G)/|\boldsymbol{r}_g - \boldsymbol{r}_G|$ is found. Finally, the center of mass of the ghost lipid $\boldsymbol{r}_{g,\text{ghost}}$ is calculated by adding twice the distance of the center of mass of the lipid to its furthest atom along the vector, i.e $\boldsymbol{r}_{g,\text{ghost}} = \boldsymbol{r}_g + 2(\boldsymbol{r}_{\max} - \boldsymbol{r}_g) \cdot (\boldsymbol{r}_g - \boldsymbol{r}_G)/|\boldsymbol{r}_g - \boldsymbol{r}_G|$. In the case of the vesicles, the ghost of inner leaflets are computed using the position of the closest atom and by removing twice the distance.

Using ghost lipids during tessellation allows MLLPA to accurately estimate the individual volume per molecule, based on the Voronoi cell[19]. For most of the geometries, MLLPA tries automatically to generate ghosts for all the types of molecules provided. However, in certain cases the generation of ghosts must be avoided, *e.g.* with molecules completely inserted

inside the membrane such as the cholesterol. For this reason, some molecule types can be specifically excluded from the ghost generation when calling *doVoro*.

The Voronoi tessellation is then computed frame by frame. The volumes and the list of neighbours of each cell of the tessellation are extracted, as well as vertices for display purposes. The neighbour list is cured twice, first by removing the ghosts from the list to only keep "real" lipids, then by removing spurious neighbours. Spurious neighbours can appear during the mathematical process of tiling the 3-Dimensional space; particles can be assigned as closest neighbours while other particles are technically in between them because the geometrical construction can find common vertices. In this case, the surface area of the face shared by the two tessellation cells will be abnormally small ($> 1\%$ of the total surface of the tessellation cell). To remove the latter, a threshold is applied to only keep neighbours with a face surface greater than 1%. The result of the tessellation is an instance of the custom class *Tessellation* designed within MLLPA.

Neighbour phase composition mapping

Once the tessellation of the system has been computed, MLLPA can combine it with the Machine Learning predictions to calculate the phase composition of the neighbours surrounding each lipid, using the tessellation results. This is illustrated in Fig. 4.

The function used for this step is *readNeighbors*. The function *readNeighbors* only takes one argument, which is the instance of the class *Tessellation* generated using the command *doVoro*. The composition of the neighbours states is given in the output as an array containing the number of neighbours found in each phase. The dimension of this array is *(number_frames, number_molecules, number_phases)*. To avoid any confusion in the attribution of the position in the last dimension of the array with the respective state, the function also outputs an array with the sorted names of the phases.

Software outputs

All the outputs described in the previous sections are internal variables that can only be read within Python, except for the model files created by *generateModel*. To allow the users

to save their results as standard files on the computer, MLLPA comes with a selection of functions, described below.

The instances of the class *System*, containing positions and states after the Machine Learning predictions, can be saved using the function *saveSystems*. The function takes as an input the list of all instances to save, but also the desired output format. *saveSystems* can save the data in 3 different formats: (i) standard text spreadsheets in *.csv file, and hierarchically structured (ii) text *.xml file or (iii) binary *.h5 file to be read with HDF5 readers. When instances are saved, the centers of mass of all molecules along with their states are saved in the file for each frame of the simulation. Since *.csv file can only support 2-dimensional arrays for input, the *(number_frames, number_molecules, number_dimensions)* centers of mass and *(number_frames, number_molecules*) phases are flattened first into arrays of dimensions *(number_frames* $\times$ *number_molecules, number_dimensions)* and *(number_frames* $\times$ *number_molecules,)* respectively.

The instances of the class *Tessellation*, containing all the information on the cell found during the tessellation along with the neighbour composition, can be saved using the function *saveVoro*. Similarly to *saveSystems*, *saveVoro* takes as an input the list of all instances to save and the desired output format. Likewise, the data can be exported as *.csv, *.xml or *.h5 formats. Because of the limits of the *.csv format, the vertices are not saved if this format is used.

VALIDATION SIMULATIONS

Methods

MLLPA was tested with various MD simulations of lipid membranes, mostly planar bilayers but also vesicles. All the systems were created using the CHARMM-GUI website[34–38]. All simulations were performed using GROMACS[39,40] and the force fields Charmm36[41] for all-atom simulations, or Martini[42–44] for the coarse grained ones. The simulations were always ran long enough to reach equilibrium, prior to collecting statistics. Unless stated, this corresponds to 25 ns of equilibration and another 25 ns of production. A complete description

of the simulation parameters used is provided in the Supplementary Materials. All visual representations of the simulations were obtained using Ovito 2.9[45].

Results

In order to assess the accuracy and stability of the software, we simulated lipid membranes composed of different types of lipids and with different geometries. Several types of tests were conducted on these membranes, which we detail in the different subsections below.

Gel/Fluid transitions in a single lipid type bilayer

We started by training MLLPA with pure lipid bilayers made of a single type of lipid: DPPC, DPPE or DMPE. For DPPC, we tested both all-atom (Charmm36) and coarse-grained force-fields (Martini). MLLPA was always set to identify either the gel or fluid phases, and was trained on at least 500 lipids in each phase. The systems were selected to be always at least 30 degrees above and below the melting transition that we found by simulating over a wide range of temperatures. The nature of the phases was confirmed by controlling the areas per lipid and tail order parameters (cf. Supplementary Materials). The total accuracy (mean accuracy over gel and fluid predictions) of the models trained using this protocol always exceeded 85%, with DPPC (all-atom) measured with an accuracy of $98.0 \pm 0.9\%$ (cf. Fig. 5(Top)). The coarse-grained model achieved a slightly lower score then its all-atom counterpart, with only $86.8 \pm 0.7\%$. Such accuracy is extremely satisfying for identification of lipid phases. Complete details of all the scores are provided in Table 2 of the Supplementary Materials.

For the distance configuration space, the lipid types simulated in the all-atom force field were all trained with a neighbour rank set to 6 and the Martini coarse-grain force field was treated with a neighbour rank of 4, although tests showed little effect of the exact rank selection for most lipids, as illustrated in Fig. 5(Bottom).

To assess the execution speed of MLLPA, we ran tests on a standard local computer from mid-2011 (Intel Core i5 2.5 GHz, AMD Radeon HD 6750M 512 MB, 20 GB 1333 MHz DDR3). During these tests, MLLPA opened simulation files with different numbers of lipid molecules,

analysed their phase, tessellated the space and listed the neighbours. The efficiency of MLLPA is constant with the number of atoms being processed, with an average of 10 μs per frame per atom processed. MLLPA was in fact found ten times less efficient when a small amount of frames is analysed than with a large trajectory, with an average speed of 33 μs.atom^{-1}.frame^{-1} for 10 frames and 4 μs.atom^{-1}.frame^{-1} for 5000 frames. This is due to MDAnalysis requiring a constant time to first open the trajectory file. Overall the speed obtained for large trajectories is extremely satisfying. Details on the speed tests are provided in the Supplementary Materials.

The models generated previously can be directly used to follow the evolution of a membrane during its melting transition. We have demonstrated this in a recently published paper, where we used MLLPA to visualise the membrane domain topography while increasing the temperature of the system[19]. We propose here to use MLLPA to produce a dynamic video rendering of a phase transition in a DPPC membrane while cooling it from 358 to 288 K (snapshot of the video given in Fig. 6, complete video available in Supplementary Materials). As demonstrated in our previous work, monitoring the membrane composition by identifying states of individual lipids together with a dynamical neighbours list enables a quantitative analysis of the local lipid environment during the phase transition).

Binary and ternary membranes

The control tests discussed so far were all performed on pure lipid bilayers. We therefore assessed how efficient MLLPA was on lipid mixtures by analysing first a binary mixture of DPPC and DPPE. These two lipids have identical chains but different headgroups and melt at different temperatures. We generated for this purpose a set of bilayers with different lipid ratio using CHARMM-GUI, ranging from 0 to 100% of DPPE. The number of lipids was kept constant equal to 64. Each of these systems was simulated at temperatures ranging from 293 to 353 K for 50 ns, and the last 25 ns of the run were analysed by MLLPA to measure the ratio of lipids found in the fluid phase. The results are shown in Fig. 7.

The gel/fluid ratios found for all systems are consistent with the known DPPC/DPPE temperature-composition phase diagram. We were indeed able to superimpose the MLLPA results with the experimental region of binary phase coexistence[46]. We performed the sim-

ulation at 328 K of a larger 1/1 system made of a total of 256 lipids to precisely measure the fluid/gel ratio of each lipid species, and found that 46 ± 8% of the DPPE were in the gel state, while only 38 ± 7 % of the DPPC were found in that state. This shows that the high melting temperature DPPE molecules are more likely to display a gel state, while the low melting temperature DPPC molecules are rather found in the fluid state. This result is consistent with the mean field picture of a coexistence between DPPC-rich liquidus and DPPE-rich solidus phases.

Confident in the accuracy of MLLPA on binary systems, we went further and tried to study a ternary membrane made of DPPC, DOPC and Cholesterol at different ratios. These mixtures known as "raft forming mixtures" are known to display a new type of ordering called "Liquid ordered phase" (Lo). The emergence of the Lo phase poses a real challenge to MLLPA, as the cholesterol-rich liquid ordered phase L_o is characterised by very ordered chain tails, similar to the gel phase[47]. Due to similarity with the gel phase, we expect that DPPC lipids in the L_o phase will mostly be assigned by MLLPA to a gel state for a MLLPA model trained with a pure DPPC system. A more complete study of the L_o phase via MLLPA is detailed in the last section to verify this aspect.

This system provides an example of a situation where some lipid components are not subjected to the lipid phase transition. DOPC for instance has such a low melting point temperature (-20°C) that no phase transition is observed in the range of simulated temperatures. We therefore prepared bilayers made of 64 lipids at different compositions, and simulated them for 50 ns at 298 K. Once again, the last 25 ns were analysed with MLLPA. This time, only the DPPC molecules had their states predicted by Machine Learning. DOPC and Cholesterol molecules were both analysed with the function *setStates*, respectively with the labels *fluid* and *cholesterol*. Since in this setup only the DPPC molecules undergo a phase transition, only the ratio of DPPC in the fluid phase over the whole number of DPPC molecules in the systems were counted. The results are given in Fig 8 (Top). A version without the coloured area can be found in the Supplementary Materials.

The different measured fluid/gel ratios match the experimental phase diagram reported for DPPC/DOPC/Cholesterol mixtures at this temperature[48]. As anticipated, MLLPA handles the L_o phase like a gel phase and does not distinguish between the presumed L_β, L_o

or L_β-L_o coexisting phases membranes, all associated with a fluid ratio circa 26% (blue domains in Fig. 8 (Top)). The predominant fluid character of DPPC molecules belonging to the DOPC-rich liquid disordered phase L_d (structurally similar to L_α) is however clear, with a fluid ratio of 86 $\pm$ 5% of fluids in mixtures associated to L_α domain. The L_β-L_α and L_β-L_α-L_o coexistence regions could be distinguished from the other domains, but couldn't be differentiated from each other, as both showed circa 70% of fluid molecules.

Nonetheless, the effect of the presence of cholesterol molecules has a significant effect on the phase composition of the DPPC molecules. Indeed, the L_d-L_o coexistence domain can easily be distinguished from the rest of the diagram, and found with a 50 $\pm$ 20% phase coexistence between gel and fluid phases.

Tracking molecules relative to the surrounding phase composition

We have already proved in our previous work[19] that MLLPA could be used to analyse the local environment of a lipid in a membrane, even during events such as a phase transition. We now test here the efficiency of the software when it comes to tracking the position of a non-phospholipid molecule within the membrane and relative to the phase composition. To do so, we considered the same ternary mixtures investigating this time the spatial distribution of the cholesterol (*cf* Fig. 8 (Bottom)).

This was achieved by calling the functions *doVoro* to map the molecules and their closest neighbours, and *ReadNeighbors* to collect the statistics regarding the surroundings of each molecule. From these analysis, we extracted only the phase assignation of the neighbours of the cholesterol molecules. This allowed us to plot a series of 2-D histograms showing the preferred location of the cholesterol in the membrane in terms of lipid phases for different lipid compositions of the membrane, as shown in Fig. 9 for the 50/30/20 and 70/10/20 systems representing respectively the gel-L_o and the L_d-L_o domains, and the Supplementary Materialsfor other systems.

The observations MLLPA allowed us to make indicate clearly that in our simulations the cholesterol molecules tends to remain preferably in environments matching the gel/fluid ratio measured by MLLPA. This means that these molecules prefer to stay at the interface between the gel and fluid domains. The snapshot presented in Fig. 8 (Bottom) suggests a

separation in two domains between a DOPC rich fluid region and a DPPC rich gel fluid, with some clustered cholesterol seemingly remaining at the interface, and a few dispersed DPPC lipids in fluid state (Fig. 9 (Bottom)). However, because our methods lacks the capability of discriminating L_o and gel phase, it is difficult to conclude on the nature of the domains visible in Fig. 8 (Bottom).

Lipid vesicle

In order to demonstrate that MLLPA is able to deal with systems more complicated than single flat bilayers, we generated via CHARMM-GUI a vesicle made of a DPPC/DOPC/Cholesterol mixture of *ca* 70/5/25. Due to the large system size, the Martini force field was used in this case. The vesicle was simulated for 1 μs at 283 K. Following the simulation, the phase composition of the molecule was measured using MLLPA at 71.7 $\pm$ 1.0% of lipids in the fluid phase. A visualisation of the outer leaflet of the vesicle, using a Lambert azimuthal equal-area projection is shown in Figure 11 of the Supplementary Materials. This is a significant difference from the previous results on Charmm and from the literature, where a bilayer made of such lipid composition would be expected to display a L_β-L_o domain coexistence. However, it has been reported in the literature that obtaining a gel phase in coarse-grained vesicles can be quite challenging[49].

Since our objective here is to test the capacity of MLLPA to analyse the average environment of a molecule of interest, here cholesterol, in a complex geometry such as a vesicle, we decided to go further with the analysis despite our vesicle not being in the expected phase. The result of the tessellation and neighbour searching algorithms was found to be perfectly similar to result obtained in a plane bilayer with a all-atom force field model. Indeed, as shown in the Supplementary Materials, the cholesterol is mainly located at the interface between the gel and fluid phase domains.

Limitations of MLLPA: L_o-gel state recognition

So far our tests have successfully proven that MLLPA is a reliable and efficient tool to identify gel and fluid phases in lipid membranes, and then to locate the average phase environment of a given lipid molecule. In this last section, we discuss one current limit of the software, which

is about the discrimination between L_o and gel phases in cholesterol-phospholipid mixtures.

Until now, we did not attempt to distinguish between the two ordered L_o and gel phases occurring in ternary raft forming mixtures. We now check whether MLLPA can reliably make a distinction among DPPC molecules pertaining to each one of these two environments. This poses a real challenge, as Fig. 10 (Top) shows, because lipids in cholesterol-rich L_o domains display typical elongated tails, very much as in the gel phase.

We therefore exposed MLLPA to three training sets: G,Lo,Ld. The gel phase training set G consisted in a pure DPPC membrane simulated at 298 K. The liquid disordered phase training set Ld was a collection of DPPC conformations arising from the simulation of a DOPC-rich membrane (DPPC/DOPC 1:9) that the previous binary gel/fluid MLLPA model found to be at $> 90\%$ in the fluid phase (*cf.* Fig. 8 (Top)). Finally, the liquid ordered phase training set Lo was taken from a DPPC/Chol 1:1 membrane, that the previous gel/fluid MLLPA model was categorizing as similar to a gel phase. As a control, a segment order parameter analysis of the Lo and G training sets was performed, which lead to a slightly different value (see Supplementary Materials). Because of differences in the lipid composition and number of DPPC molecules, the number of frames analysed for each training simulations sets was adjusted to give to the three training sample sets an identical size. Complete details of all the scores are provided in Table 3 of the Supplementary Materials.

The first critical observation is that the training scores generated by MLLPA seem to indicate that MLLPA is perfectly able to recognise the L_o phase from both the gel and fluid phases. Indeed, the total score achieved was of 98.8 ± 0.4 %, with an accuracy of 98 ± 1 % and 98.5 ± 1.0 % respectively for the gel and L_o lipids.

However, we can see that some ML algorithms have trouble to differentiate gel from L_o. For example, the SVM algorithm on coordinates, which achieves an accuracy ¿ 90% to recognise fluid lipids, found both gel and L_o with an accuracy circa 60%, close to an uninformed guess. This leads to serious concern about the reliability of the scores obtained for L_o. To investigate this further, we performed a cross-check analysis by training the software with one system and making it analyze the other system(s) as well as itself. Some significant results are shown in Fig. 10 (Bottom), and all the detailed scores and results are given in the Table 4 of the Supplementary Materials.

When analysed with a model trained on two states gel/fluid, the gel system is naturally found with a gel ratio of $97 \pm 2\%$ consistent with the training scores. However, when analysed with the model trained on the three states, the ratio of gel found in the pure low temperature DPPC (gel) system drops to 75 ± 8 % and 22% of the lipid are identified as being part of the L_o phase. Since the L_o phase implies the presence of cholesterol in the membrane, the pure DPPC bilayer should not display such an amount of Lo phase. Moreover, the analysis by the three states model of the 1:1 DPPC/Chol system (L_o) only predicts a L_o state ratio of 66 ± 12 %. This score is only marginally higher than a L_o-gel random guess, proving that MLLPA cannot recognise unambiguously the L_o phase in spite of the latter being presumably dominant. This shows that there is definitely an important cross-talk between the gel and L_o conformations, that our current ML analysis is not able to significantly suppress.

There are two possible explanations regarding the impossibility of discriminating liquid ordered from pure gel. First, it could be that our features spaces (distances and coordinates) both miss some important characteristics that would discriminate the states at the single DPPC molecular level. For instance, the construction of our features spaces requires the lipids to be oriented according to their tensor of gyration. This process removes all information on the tilt of the chain relative to the bilayer normal. As a consequence, if two phases can only be differentiated by this tilt, MLLPA will not be able to distinguish between them. The second possibility is that single L_o and gel molecular states are so similar in reality that only the proximity of a cholesterol molecules can tell which of the two states is actually realised. This last result shows a limitation of MLLPA as far as recognizing the lipid phase at the single lipid molecule level is concerned. We can however foresee this limitation as an important analysis tool, as a failure from MLLPA to distinguish between two phases would demonstrate the similarity of their intrinsic configuration.

SUMMARY

We introduced a robust and predictive ML tool, MLLPA, that can efficiently assign a state to a lipid based on its likely participation in a gel or fluid phase. We have proven with these simulations that MLLPA was a robust and efficient analysis software that could deal with

many different models and types of lipids. The accuracy in identifying the gel or fluid state is really satisfying for a Machine Learning-based software, with the most common types of lipid achieving scores above 95%. This is essential for the capacity of MLLPA to detect the average environment of membrane molecules, as shown with the tracking of cholesterol molecules in the membrane. All of these operations can be achieved at reasonable speed, with an average processing time of 3 μs/atom/frame on a standard local computer without any parallelisation.

We expect that MLLPA has vocation to contribute significantly to the analysis of phase-driven interactions in lipid membranes such as hydrophobic pollutants[50–52], or to analysis of membrane proteins local environment[53,54].

We have also shown in this paper a limit of the software, notably in the case of cholesterol induced lipid chain order, where no conclusion could be reached based solely on the molecular conformation of the DPPC molecules. Future work should indicate which way might lead to the best possible discrimination of the liquid ordered-gel domains.

ACKNOWLEDGMENTS

V. W. warmly thanks Adrien Gola for helping with Ovito set-up and conversion scripts. The authors gratefully acknowledge support from the high performance cluster (HPC) Equip@Meso from the University of Strasbourg, through grant no. G2019A131C, and use of the research computing facility at King's College London, Rosalind (https://rosalind.kcl.ac.uk).

Additional Supporting Information may be found in the online version of this article.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Zenodo at http://doi.org/10.5281/zenodo.4300706. All source files of the software are openly available in GitHub at https://github.com/vivien-walter/mllpa. Documentation and tutorials are available at https://vivien-walter.github.io/mllpa.

References

1. G. Cevc and D. Marsh, eds., *Phospholipid bilayers physical principles and models* (John Wiley & Sons, 1987).

2. L. A. B. Ole G. Mouritsen, *LIFE - AS A MATTER OF FAT* (Springer-Verlag GmbH, 2015).

3. S. Mabrey and J. M. Sturtevant, PNAS **73**, 3862 (1976).

4. T. Heimburg, *Thermal biophysics of membranes*, Tutorials in biophysics (Wiley-VCH, Weinheim, 2007), ISBN 9783527404711, literaturverz. S. 339 - 351.

5. D. Marsh, *Handbook of Lipid Bilayers* (CRC Press, Boca Raton, 2013), 2nd ed.

6. J. F. Nagle, The Journal of Chemical Physics **58**, 252 (1973), ISSN 0021-9606.

7. D. Marsh, The Journal of Membrane Biology **18**, 145 (1974).

8. S. T. Marcelja, Biochimica et Biophysica Acta - Biomembranes **367**, 165 (1974).

9. M. J. Zuckermann and O. G. Mouritsen, European Biophysics Journal **15**, 77 (1987), 10.1007/BF00257501.

10. H. I. Petrache, K. Tu, and J. F. Nagle, Biophysical Journal **76**, 2479 (1999).

11. H. I. Petrache, S. W. Dodd, and M. F. Brown, Biophysical Journal **79**, 3172 (2000).

12. E. Cubuk, S. Schoenholz, J. Rieser, B. Malone, J. Rottler, D. Durian, E. Kaxiras, and A. Liu, Physical Review Letters **114**, 108001 (2015).

13. J. Carrasquilla and R. G. Melko, Nature Physics **13**, 413 (2017).

14. T. C. Le and N. Tran, ACS Applied Nano Materials **2**, 1637 (2019).

15. C. A. Lopez, V. V. Vesselinov, S. Gnanakaran, and B. S. Alexandrov, Journal of Chemical Theory and Computation **15**, 6343 (2019).

16. S. S. Iyer, A. Negi, and A. Srivastava, Journal of Chemical Theory and Computation **16**, 2736 (2020).

17. M. Aghaaminiha, S. A. Ghanadian, E. Ahmadi, and A. M. Farnoud, Biochimica et biophysica acta. Biomembranes **1862**, 183350 (2020).

18. T. E. de Oliveira, F. Leonforte, L. Nicolas-Morgantini, A.-L. Fameau, B. Querleux, F. Thalmann, and C. M. Marques, Physical Review Research **2**, 013075 (2020).

19. V. Walter, C. Ruscher, O. Benzerara, C. M. Marques, and F. Thalmann, Physical chemistry chemical physics : PCCP **22**, 19147 (2020), ISSN 1463-9084.

20. D. Sánchez-Gutiérrez, M. Tozluoglu, J. D. Barry, A. Pascual, Y. Mao, , and L. M. Escudero, The Embo Journal **35**, 77 (2016).

21. G. Voronoi, Journal für die reine und angewandte Mathematik **133**, 97 (1908).

22. G. Voronoi, Journal für die reine und angewandte Mathematik **134**, 198 (1908).

23. W. J. Allen, J. A. Lemkul, and D. R. Bevan, Journal of Computational Chemistry **30**, 1952 (2008).

24. S. Buchoux, Bioinformatics **33**, 133 (2017).

25. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Journal of Machine Learning Research **12**, 2825 (2011).

26. C. H. Rycroft, Chaos: An Interdisciplinary Journal of Nonlinear Science **19**, 041111 (2009).

27. W. Smith, *Tess*, URL `https://github.com/wackywendell/tess`.

28. N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, Journal of Computational Chemistry **32**, 2319 (2011).

29. R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, J. Domanski, D. L. Dotson, S. Buchoux, I. M. Kenney, et al., Proceedings of the 15th Python in Science Conference pp. 98–105 (2016).

30. GitHub repo, URL `github.com/vivien-walter/mllpa`.

31. Zenodo repo, URL `DOI: 10.5281/zenodo.4300706`.

32. MDAnalysis - Trajectory Readers, URL `https://docs.mdanalysis.org/stable/documentation_pages/coordinates/init.html`.

33. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., Nature **585**, 357 (2020).

34. S. Jo, T. Kim, V. Iyer, , and W. Im, Journal of Computational Chemistry **29**, 1859 (2008).

35. B. Brooks, C. B. III, J. A.D. MacKerell, L. Nilsson, R. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, et al., Journal of Computational Chemistry **30**, 1545 (2009).

36. J. Lee, X. Cheng, J. Swails, M. Yeom, P. Eastman, J. Lemkul, S. Wei, J. Buckner, J. Jeong, Y. Qi, et al., Journal of Chemical Theory and Computation **12**, 405 (2016).

37. E. Wu, X. Cheng, S. Jo, H. Rui, K. Song, E. Dávila-Contreras, Y. Qi, J. Lee, V. Monje-Galvan, R. Venable, et al., Journal of Computational Chemistry **35**, 1997 (2014).

38. S. Jo, J. B. Lim, J. B. Klauda, and W. Im, Biophysical Journal **97**, 50 (2009).

39. H. J. C. Berendsen, D. van der Spoel, and R. van Drunen, Computer Physics Communications **91**, 43 (1995).

40. M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C.Smith, B. Hessa, and E. Lindahl, SoftwareX **1-2**, 19 (2015).

41. R. B. Best, X. Zhu, J. Shim, P. E. M. Lopes, J. Mittal, M. Feig, , and A. D. MacKerell, Journal of Chemical Theory and Computation **8**, 3257 (2012).

42. S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, The Journal of Physical Chemistry B **111**, 7812 (2007).

43. Y. Qi, H. I. Ingólfsson, X. Cheng, J. Lee, S. J. Marrink, and W. Im, Journal of Chemical Theory and Computation **11**, 4486 (2015).

44. P. Hsu, B. M. H. Bruininks, D. Jefferies, P. C. T. de Souza, J. Lee, D. S. Patel, S. J. Marrink, Y. Qi, S. Khalid, and W. Im, Journal of Computational Chemistry **38**, 2354 (2017).

45. A. Stukowski, Modelling and Simulation in Materials Science and Engineering **18**, 015012 (2009).

46. E. J. Shimshick and H. M. McConnell, Biochemistry **12**, 2351 (1973).

47. J. H. Ipsen, G. Karlström, O. G. Mouritsen, H. Wennerström, and M. J. Zuckermann, Biochimica et Biophysica Acta - Biomembranes **905**, 162 (1987).

48. S. L. Veatch, O. Soubias, S. L. Keller, , and K. Gawrisch, PNAS **104**, 17650 (2007).

49. D. Stelter and T. Keyes, Soft Matter **15**, 8102 (2019).

50. M. D. Franova, I. Vattulainen, and O. Ollila, Biochimica et Biophysica Acta - Biomembranes **1838**, 1406 (2014).

51. J. H. Crowe, F. A. Hoekstra, K. H. N. Nguyen, and L. M. Crowe, Biochimica et Biophysica Acta - Biomembranes **1280**, 187 (1996).

52. G. Rossi and L. Monticelli, Journal of Physics: Condensed Matter **26**, 503101 (2014).

53. S. Marcelja, Biochimica et Biophysica Acta - Biomembranes **455**, 1 (1976).

54. T. Gil, J. H. Ipsen, O. G. Mouritsen, M. C. Sabra, M. M. Sperotto, and M. J. Zuckermann, Biochimica et Biophysica Acta - Biomembranes **1376**, 245 (1998).

Figure 1: Snapshots of two DPPC molecules extracted from a membrane simulated (Left) in the gel phase and (Right) in the fluid phase. The typical conformation of a lipid in the gel phase is easily seen here, with elongated tails along the normal to the plane of the membrane, while a lipid in the fluid phase have disordered tails with random orientations.

Figure 2: Illustration of the pairs between atoms along the same chain, starting from the atom in black and for different ranks as defined in MLLPA. The colour code is used to highlight the different pairs found with the same rank.

Figure 3: Illustration of the issues that can occur in (a) vesicles and (b) bilayers while assigning neighbors using tessellation; respectively by assigning neighbors through a block of solvent or through the PBC of the simulation box. Figures (c) and (d) show how the presence of ghosts corrects the issues in neighbors listing. Figures (e) and (f) illustrates how ghosts are created. The center of mass of the membrane is shown as a blue point in the vesicle and a dashed line with the bilayer. Vectors used for the creation of the ghosts are shown as coloured arrow, blue vector corresponds to $\boldsymbol{r}_g - \boldsymbol{r}_G$ while red vector is $\boldsymbol{r}_{g,\text{ghost}} - \boldsymbol{r}_G$. The examples of issues were exaggerated in this illustration for clarity.

Figure 4: The phase composition of the neighbours of a molecule is extracted according to a Voronoi tessellations. Once the list of neighbours is obtained, their states/phases are collected and analysed.

Figure 5: (Top) Training score accuracy obtained by training MLLPA to differentiate gel and fluid phases in different lipids bilayer models (C.: Charmm36 all-atom force field, M.: Martini coarse-grained force-field. The difference in model is also highlighted by the change of colour). (Bottom) Evolution of the overall accuracy of MLLPA to recognise gel and fluid phases in a DPPC bilayer as a function of the neighbour rank used for the data collection, for atomic and coarse-grained models.

Figure 6: (Top) Snapshot of the top leaflet of a DPPC bilayer undergoing phase separation during a phase transition, with a Voronoi tessellation highlighting the distinct gel (blue) and fluid (red) domains. Dashed lines show the limits of the PBC box. (Bottom) Evolution of the ratio of DPPC in the fluid phase in the bilayer during a gentle cooling, going from 358 to 288 K by decreasing the system temperature by 1 K/ns. The evolution is compared with the one of the area per lipid (in blue).

Figure 7: Phase diagrams showing the ratio of (Top) DPPC and (Bottom) DPPE lipid molecules found in the fluid phase in a 64 lipid bilayer made of different ratio of DPPE and simulated at different temperatures. Each point represents the result of a simulation run. The plain black lines are the experimental phase diagram as found by Shimshick and McConnell[46].

Figure 8: (Top) Phase diagram of ternary membranes made of DPPC/DOPC/Cholesterol at different ratios and simulated at 298 K. Each point represents the result of a simulation run. The plain and dashed black lines are the experimental phase diagram as found by Veatch *et al.* at the same temperature[48]. The colored areas highlight the average values found inside each domain (Bottom) Snapshot of a ternary membrane made of DPPC/DOPC/Cholesterol at the ratio 50/30/20 and simulated at 298 K. The DPPC are shown in red or blue depending on their states as identified by MLLPA, respectively fluid and gel. The DOPC molecules are shown in orange/dark yellow as they are always identified by MLLPA has being fluid. The cholesterol molecules, which also do not have a transition are shown in black.

Figure 9: Color coded histograms of the (# gel, # fluid) distribution of probability to find a cholesterol molecule in a environment composed of a given amount of gel and fluid lipids measured in bilayers made of DPPC/DOPC/Cholesterol mixtures of (Top) 50/30/20 and (Bottom) 70/10/20. The red dashed lines shows the measured gel/fluid ratio in all lipids (DPPC + DOPC).

Figure 10: (Top) Snapshot of membrane slices found in the disordered liquid phase packed with cholesterol (gray). Lipids have been coloured in order to highlight the different leaflet in which they are located. (Bottom) Ratio of lipids in different phases measured in two DPPC/DOPC/Cholesterol systems at 298 K: a bilayer in the gel phase and another one in the disordered liquid phase L_o. The analysis was made with a model trained on three sets of gel, L_o and fluid lipids (red). Error bars are the standard deviation on the ratio measured over a 25 ns simulation run.

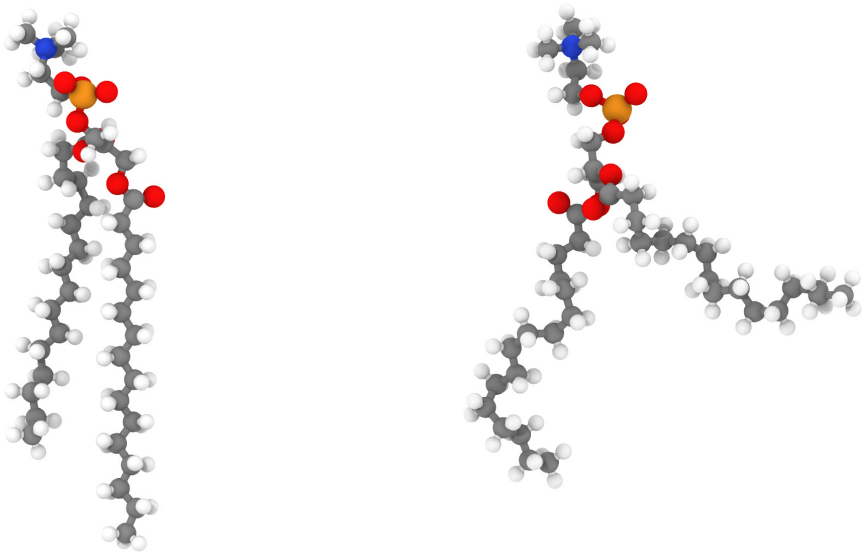


Figure 1

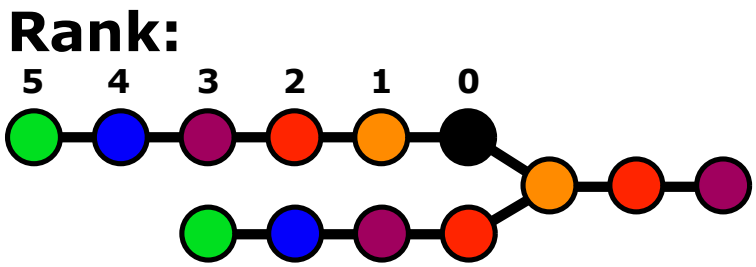


Figure 2

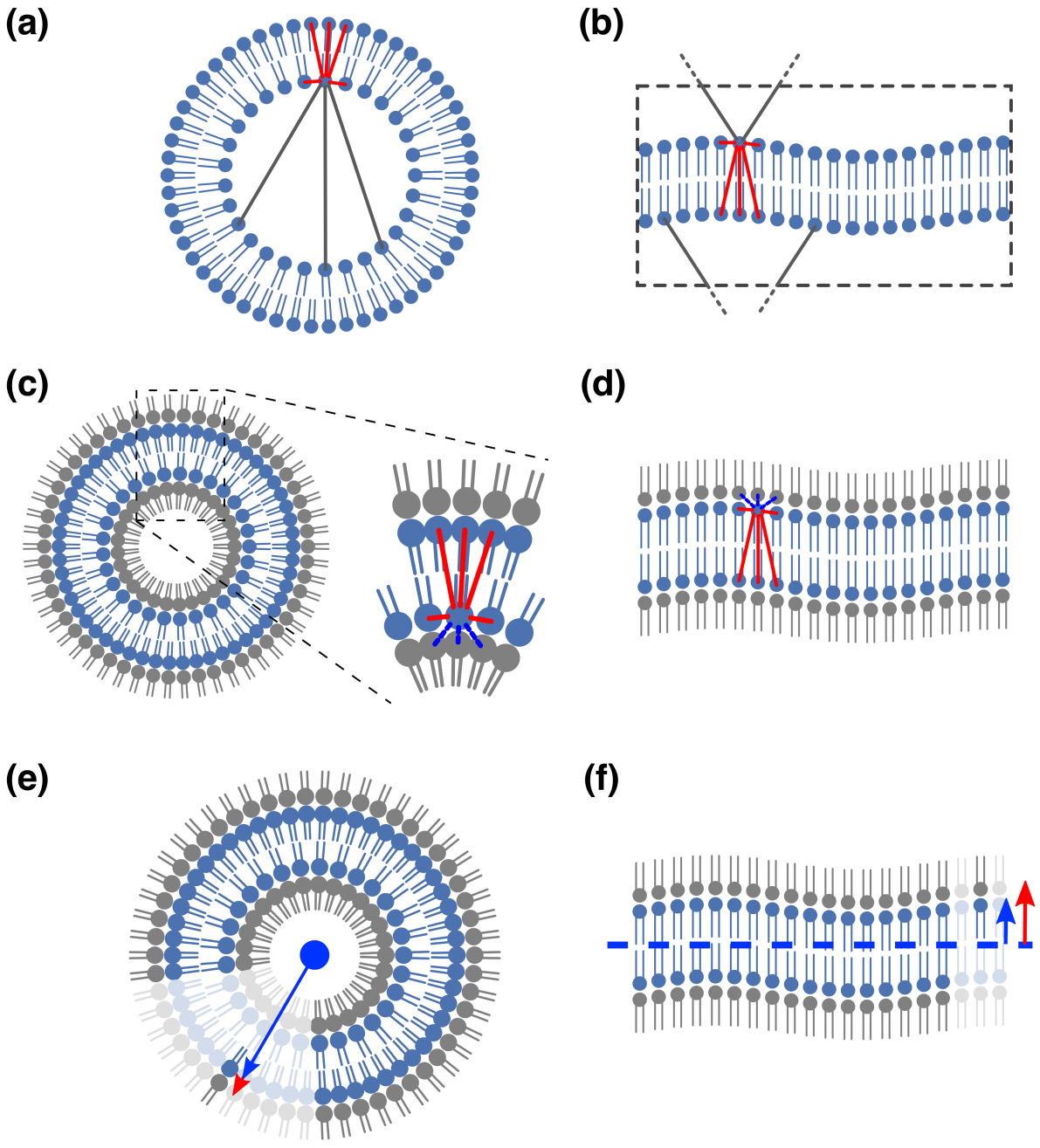


Figure 3

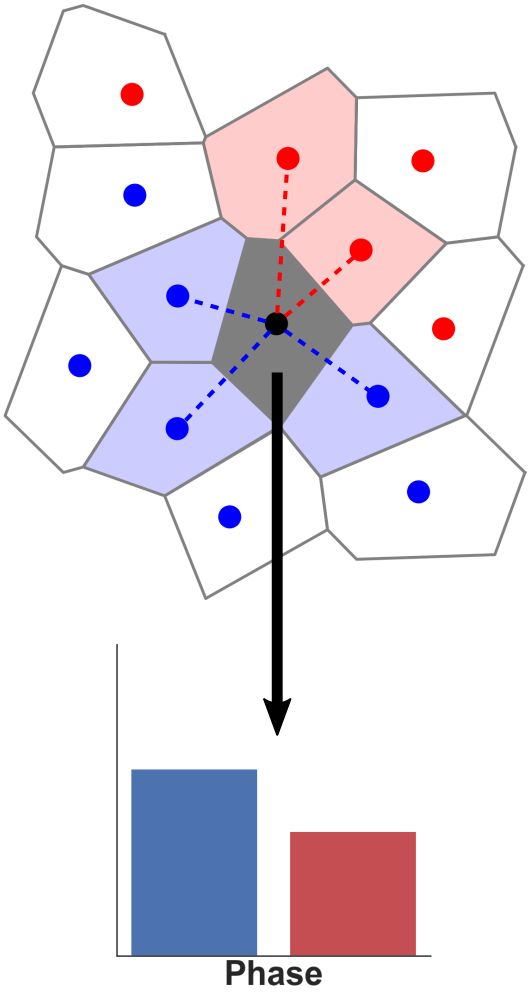


Figure 4

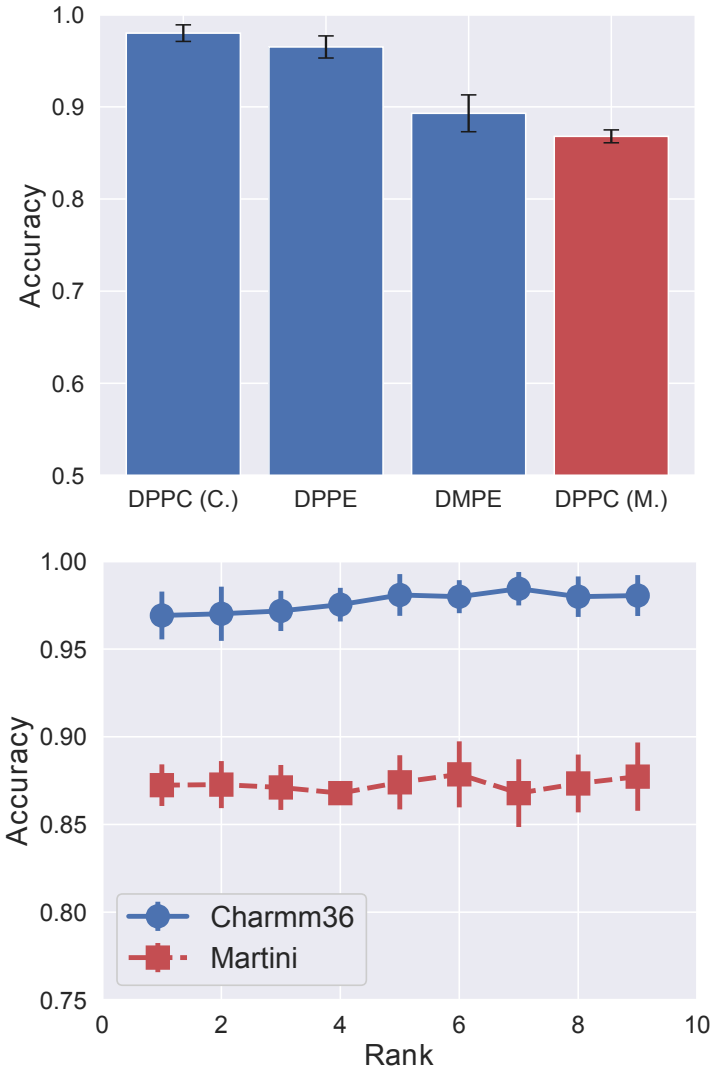

1.0
0.9
0.8
0.7
0.6
0.5
Accuracy
DPPC (C.)
DPPE
DMPE
DPPC (M.)
1.00
0.95
0.90
0.85
0.80
0.75
Accuracy
Charmm36
Martini
0
2
4
6
8
10
Rank

Figure 5

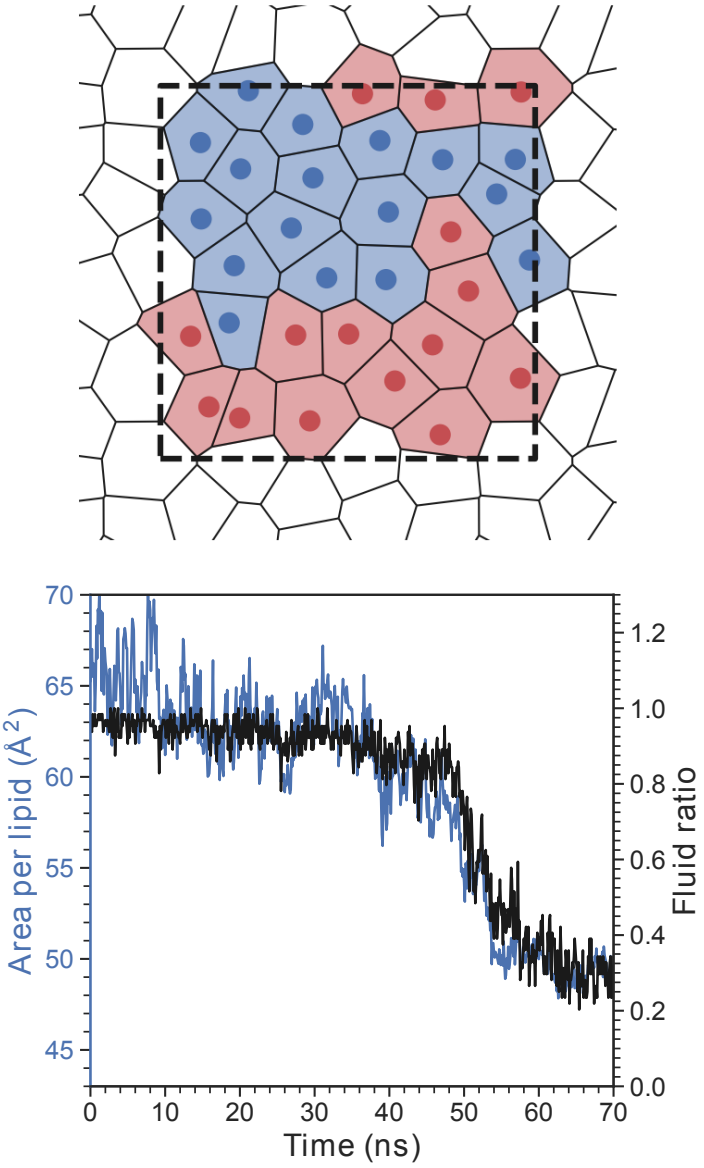

70
65
60
55
50
45
Area per lipid (Å^2)
0
10
20
30
40
50
60
70
Time (ns)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Fluid ratio

Figure 6

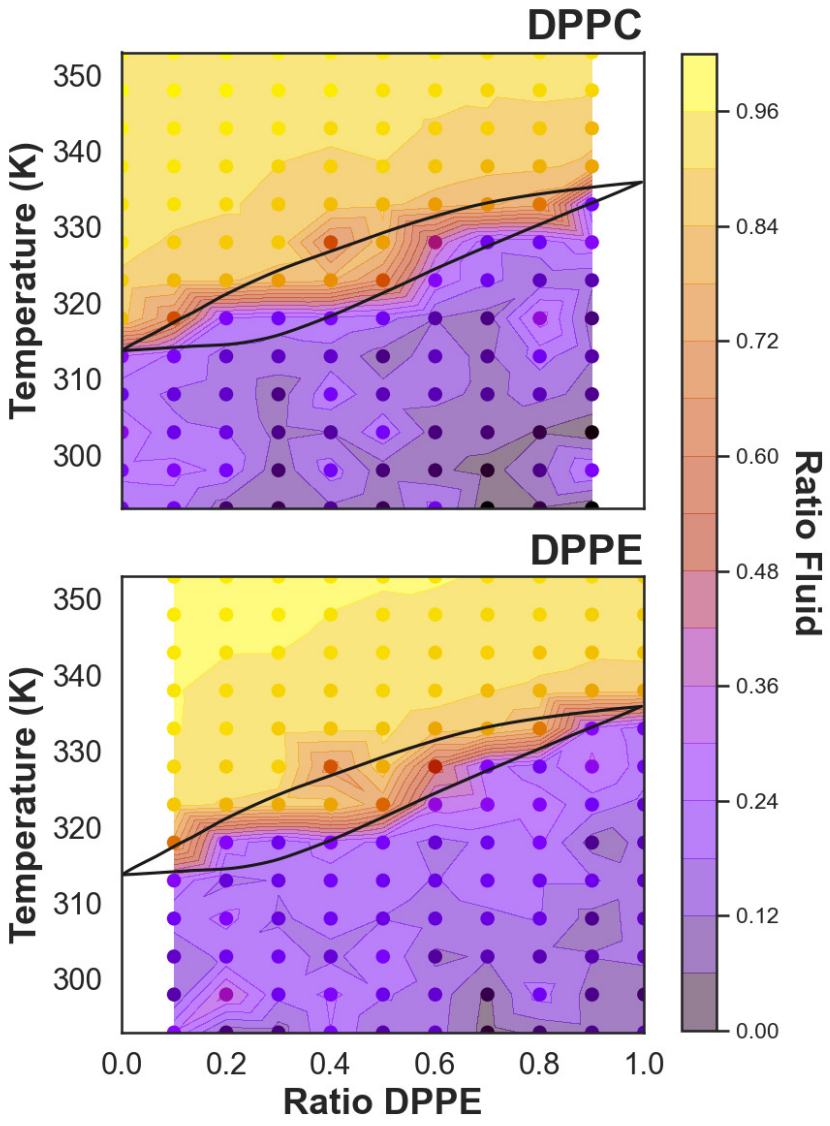
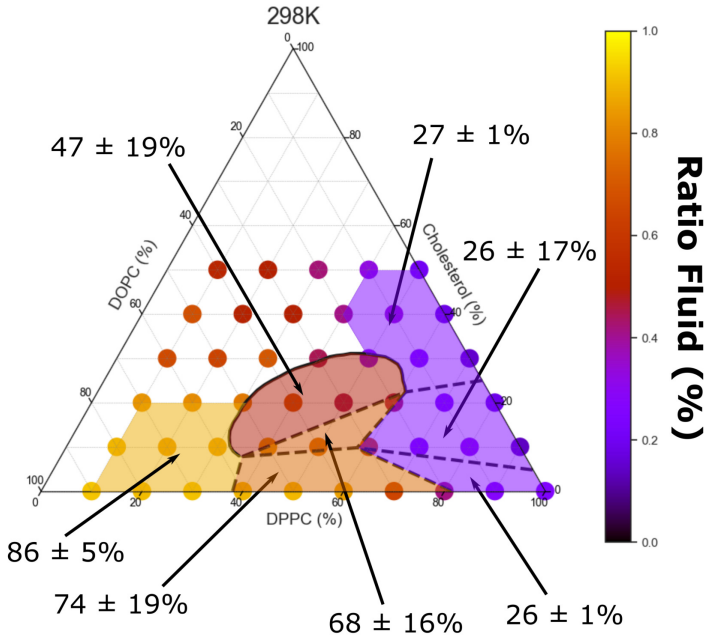

DPPC
DPPE
Temperature (K)
350
340
330
320
310
300
0.0
0.2
0.4
0.6
0.8
1.0
Ratio DPPE
Ratio Fluid
0.96
0.84
0.72
0.60
0.48
0.36
0.24
0.12
0.00

Figure 7


298K
47 ± 19%
27 ± 1%
26 ± 17%
86 ± 5%
74 ± 19%
68 ± 16%
26 ± 1%
DOPC (%)
Cholesterol (%)
DPPC (%)
Ratio Fluid (%)

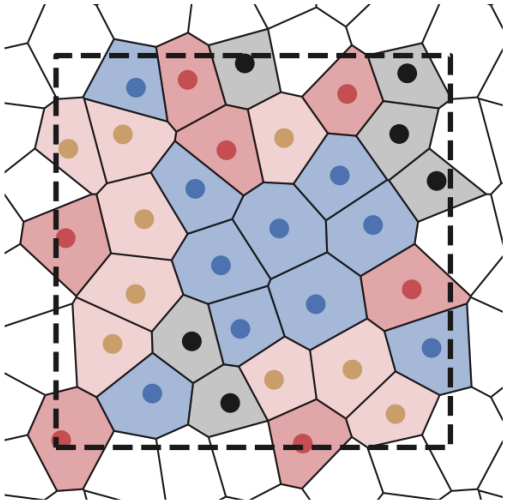


Figure 8

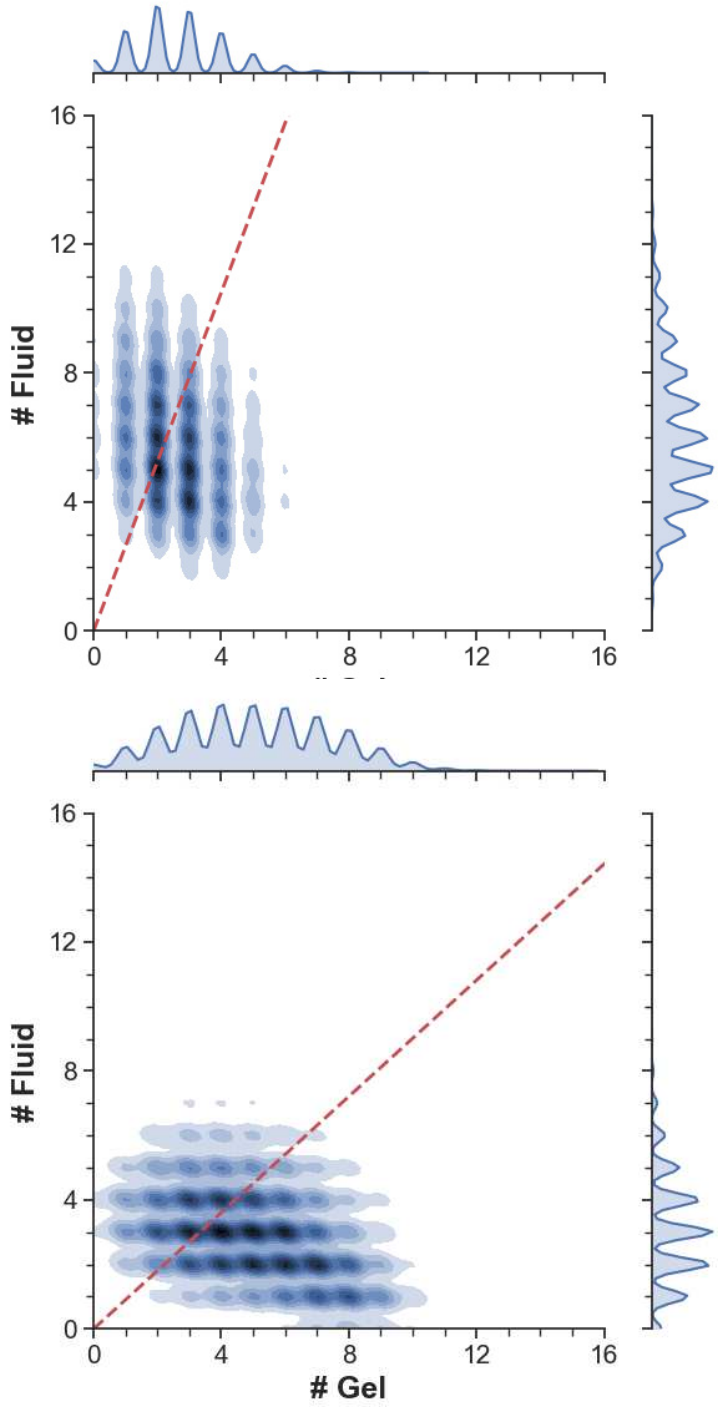


Figure 9

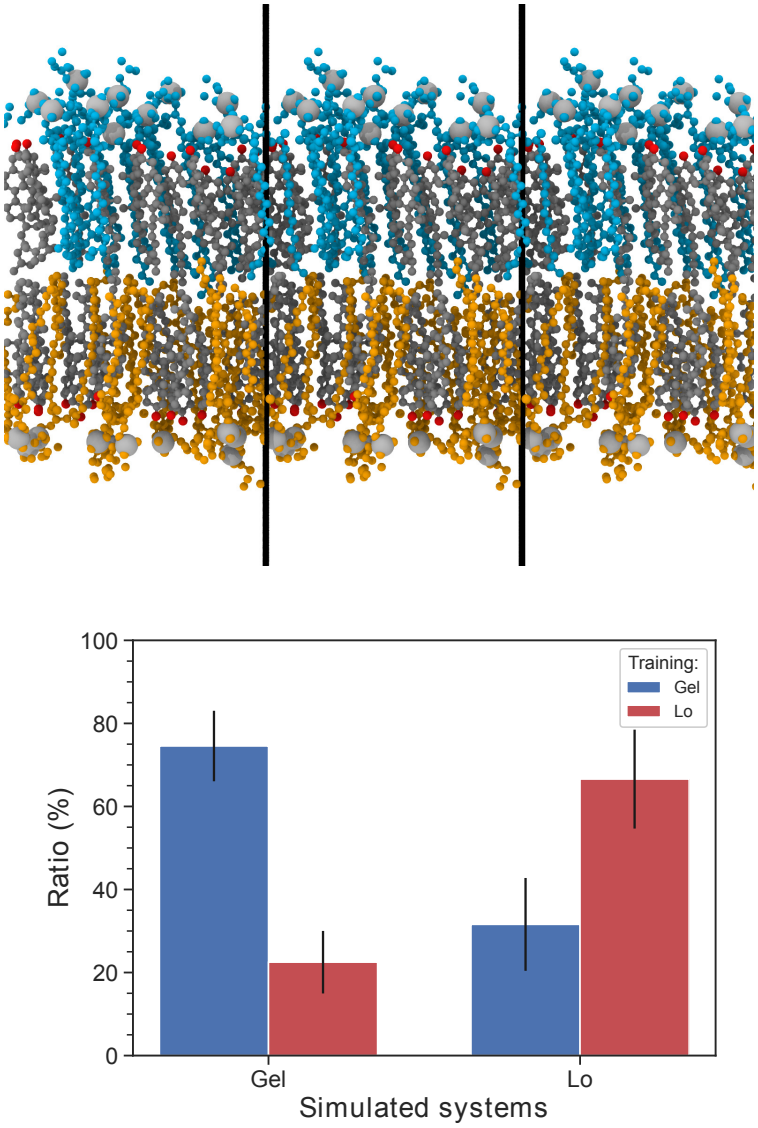

100
80
60
40
20
0
Ratio (%)
Training:
Gel
Lo
Gel
Lo
Simulated systems

Figure 10

MLLPA: A Machine Learning-assisted Python module to study phase-specific events in lipid membranes - Supplementary Materials -

Vivien Walter[*], Céline Ruscher [†], Olivier Benzerara[†],Fabrice Thalmann[†]

January 28, 2021

[*]Department of Chemistry, King's College London, Britannia House, 7 Trinity Street, London, UK

[†]Institut Charles Sadron, CNRS and University of Strasbourg, 23 rue du Loess, F-67034 Strasbourg, France

1 Description of the software

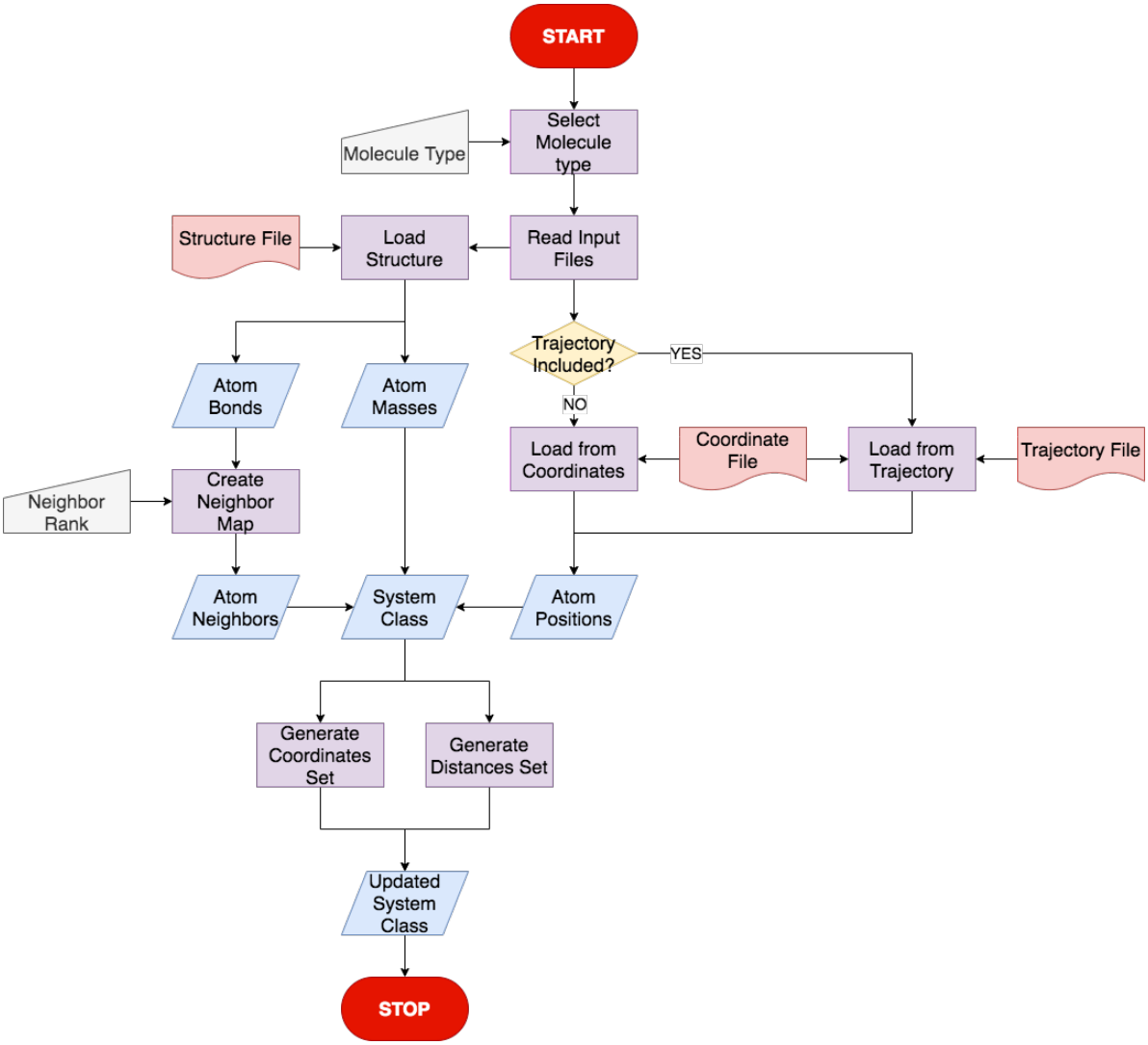


Figure 1: Flowchart of the logic structure used in MLLPA to read the simulation files and generate the instance of system class containing all the information on the selected lipid type.

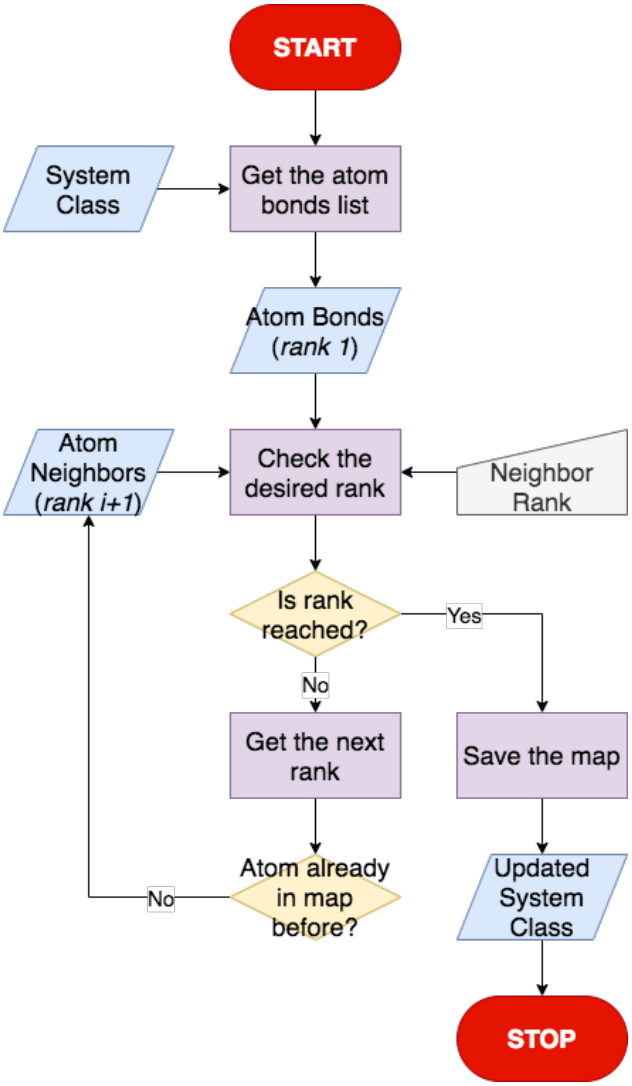


Figure 2: Flowchart of the logic structure used in MLLPA to construct the map at the neighbour rank N.

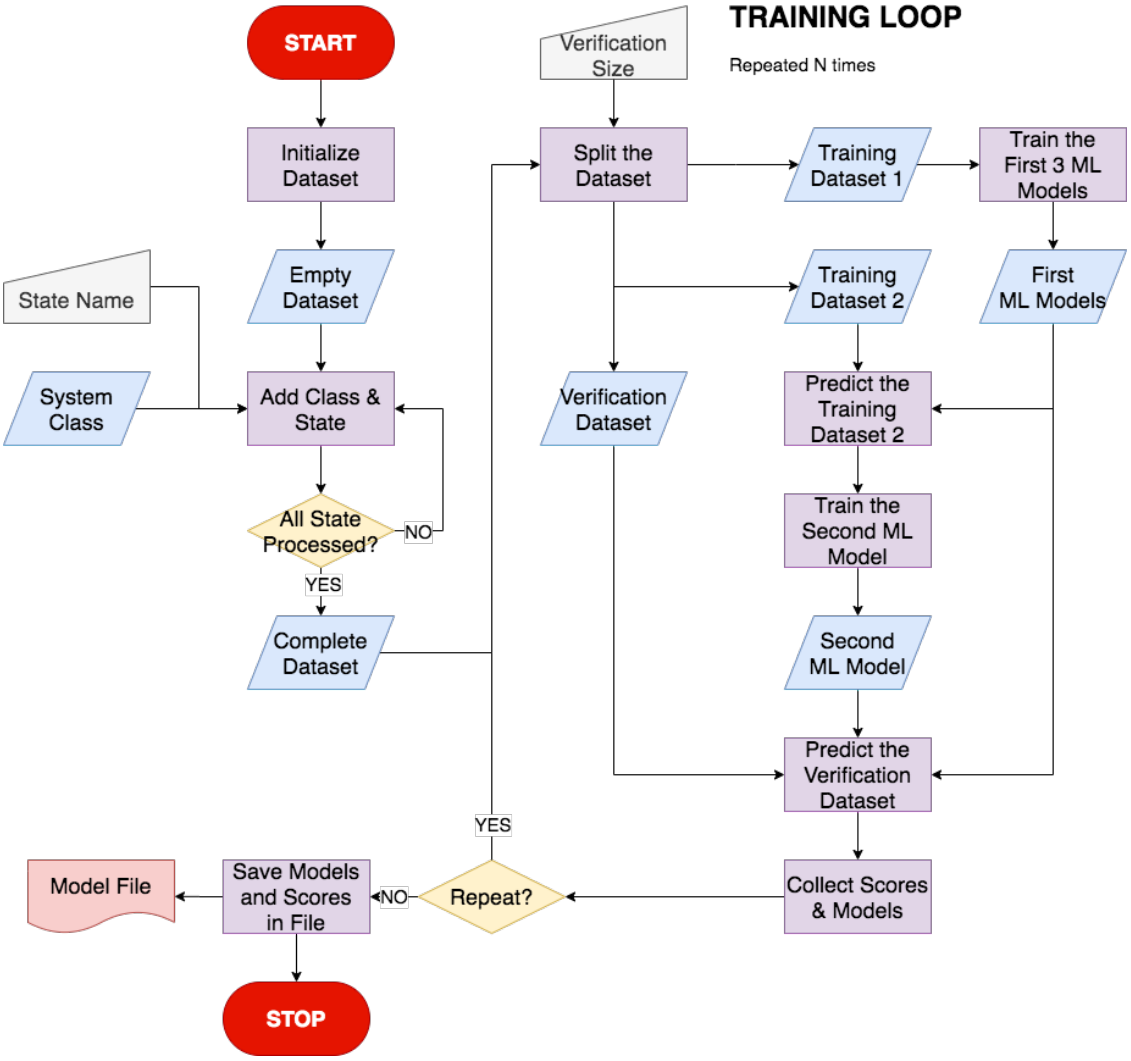


Figure 3: Flowchart of the logic structure used in MLLPA to prepare the training sets and train the Machine Learning models to predict the states of molecules.

2 Verification simulations

2.1 Simulation paramaters

The simulations on the all-atom forcefield Charmm36 (version used: June 2015) were performed on Gromacs 2016.4. The complete details on the simulation parameters are exactly the same as the ones described in our previous paper[1]. Briefly, the membranes were first relaxed through a steepest descent energy minimization, followed by a 10 ps NVT thermalization at the desired temperature. Then, the bilayer was subject to a 1 ns NPT equilibration run coupled with a semi-isotropic barostat (1 bar in all directions). The system was then further equilibrated at the desired temperature with the same semi-isotropic barostat during a second NPT equilibration step of 10 ns. Molecular dynamics production runs of 50 ns were finally generated at the same temperature and with the same semi-isotropic barostat. The analysis were performed on the last 25 ns of simulations. All time steps were set to 2 fs.

The simulation on the coarse-grain forcefield Martini (version used: 2.2) were performed on Gromacs 2019.4. For these systems, the minimisation and equilibration used were the ones provided by CHARMM-GUI[2,3] instead of the one used for the all-atom forcefield. Once the systems were ready, molecular dynamics production runs of 100 ns were generated at the required temperature. Similarly to the all-atom systems, the analysis were performed on the last 25 ns of simulations. The leap-frog integration algorithm[4] was used for the simulation runs, with a time step of 20 fs. Temperature and pressure were kept constant using respectively a V-rescale thermostat (correlation time $\tau_T = 1$ ps) and a Parrinello-Rahman semi-isotropic barostat (correlation time $\tau_P = 12$ ps, compressibility 3×10^{-4} bar^{-1}). The lipid membrane and the solvent were separated into two groups for the thermostat. All other parameters are given in Table 1.

Parameter	**Value**
cutoff-scheme	Verlet
nstlist	20
ns_type	grid
pbc	xyz
verlet-buffer-tolerance	0.005
epsilon_r	15
coulombtype	reaction-field
rcoulomb	1.1
vdw_type	cutoff
vdw-modifier	Potential-shift-Verlet
rvdw	1.1

Table 1: Names and values of the Gromacs input parameters for the MD simulation runs made with the Martini forcefield.

2.2 Speed Tests

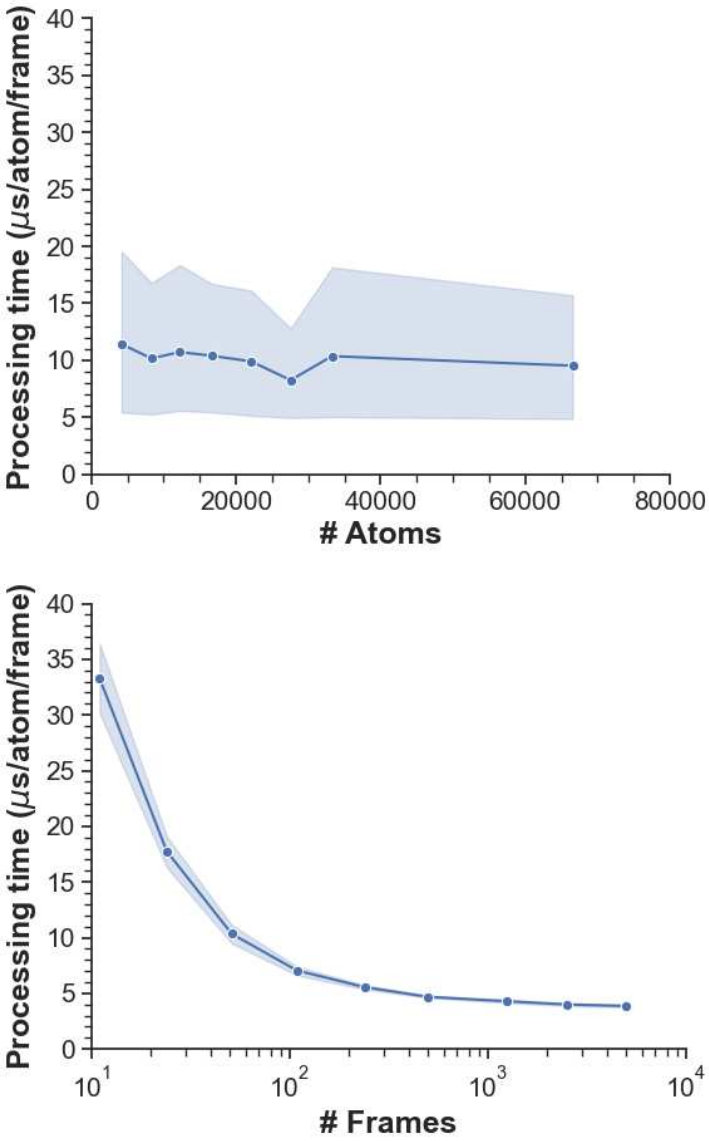


Figure 4: Evolution of the average processing time, given per atom and per frame, analysed of MLLPA (Top) as a function of the number of atoms and (Bottom) as a function of the number of frames processed. The blue area corresponds to the standard deviation measured during the speed test.

2.3 Gel/Fluid transitions

2.3.1 DPPC - Charmm

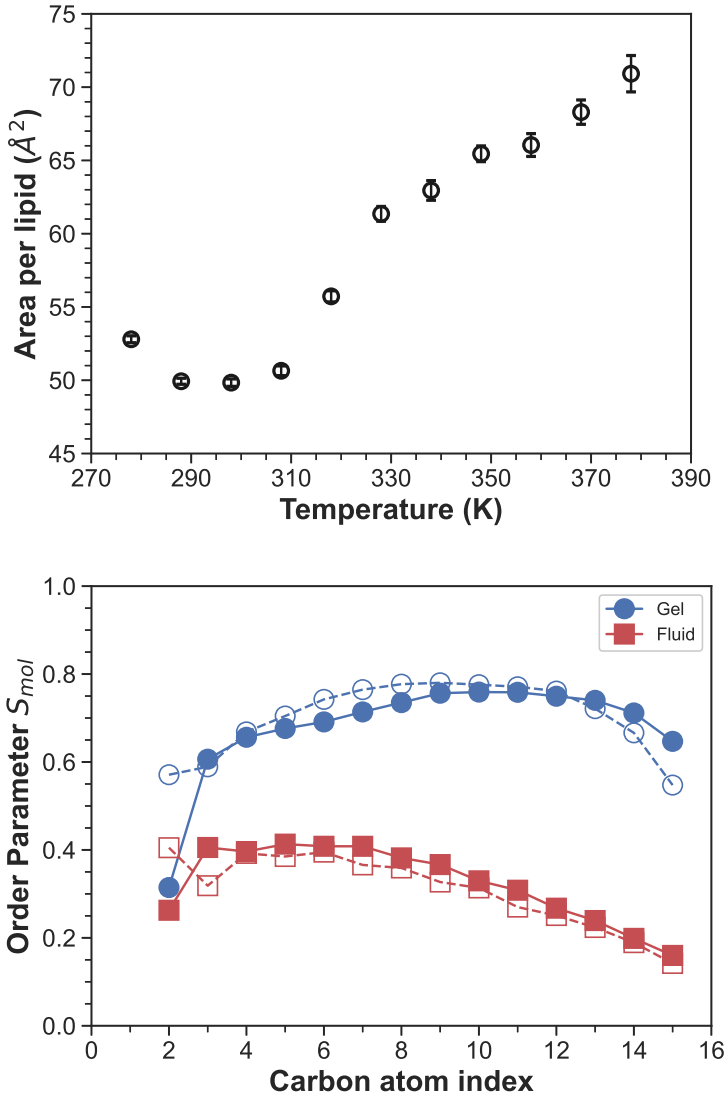


Figure 5: Measurements used to confirm that the DPPC systems simulated underwent a phase transition and could be used to train MLLPA. (Top) Evolution of the area per lipid of the system with the temperature, and (Bottom) tail order parameters measured for the system found in gel (288 K) and fluid (358 K). The order parameters have been averaged over the number of lipids and frames in the trajectory. Plain and open symbols are respectively the sn1 and sn2 tails of the lipids.

2.3.2 DMPE

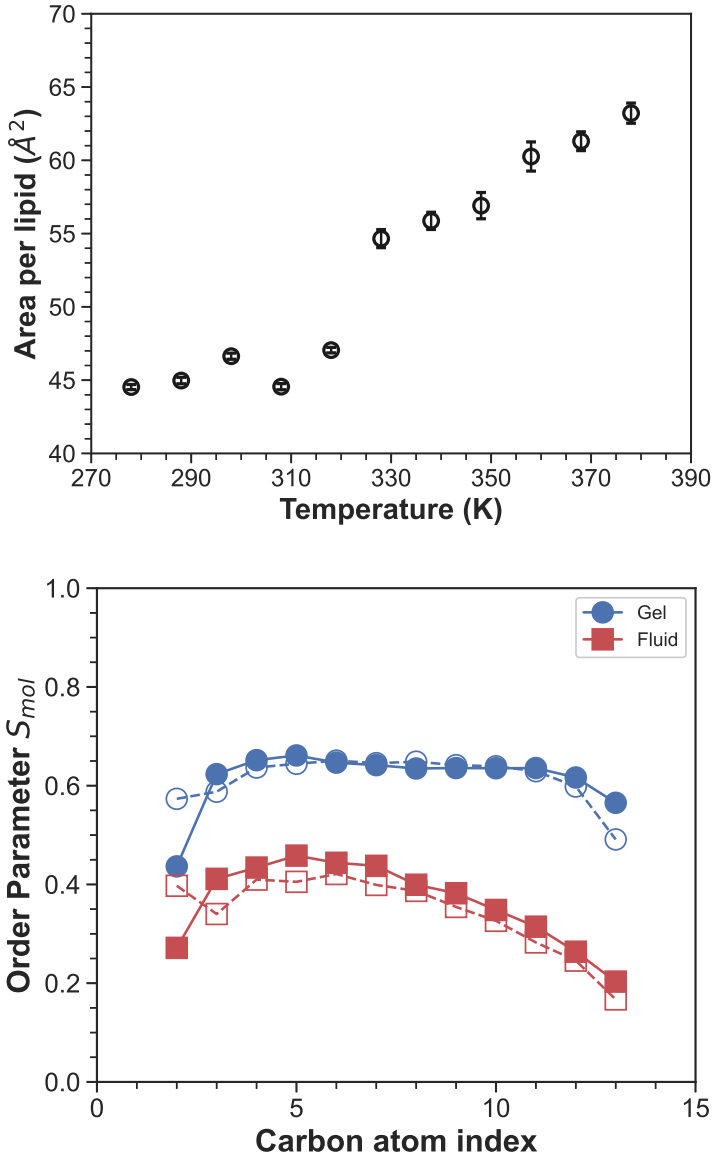


Figure 6: Measurements used to confirm that the DMPE systems simulated underwent a phase transition and could be used to train MLLPA. (Top) Evolution of the area per lipid of the system with the temperature, and (Bottom) tail order parameters measured for the system found in gel (288 K) and fluid (358 K). The order parameters have been averaged over the number of lipids and frames in the trajectory. Plain and open symbols are respectively the sn1 and sn2 tails of the lipids.

2.3.3 DPPE

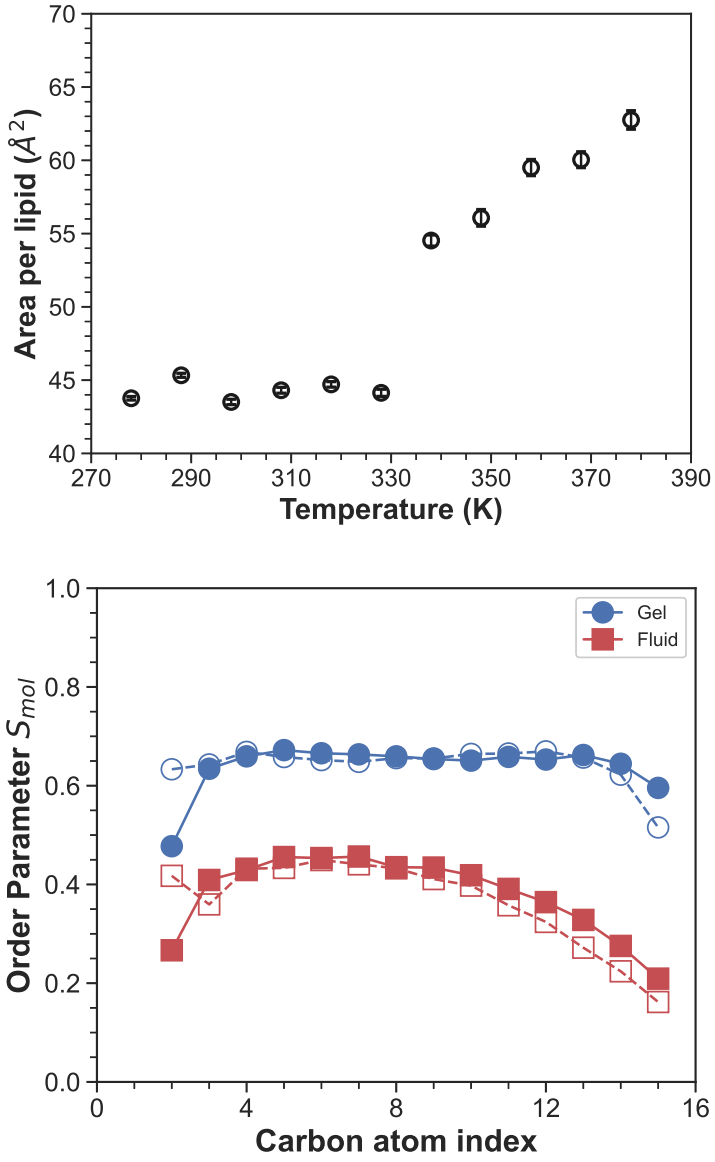


Figure 7: Measurements used to confirm that the DPPE systems simulated underwent a phase transition and could be used to train MLLPA. (Top) Evolution of the area per lipid of the system with the temperature, and (Bottom) tail order parameters measured for the system found in gel (298 K) and fluid (368 K). The order parameters have been averaged over the number of lipids and frames in the trajectory. Plain and open symbols are respectively the sn1 and sn2 tails of the lipids.

2.3.4 DPPC - Martini

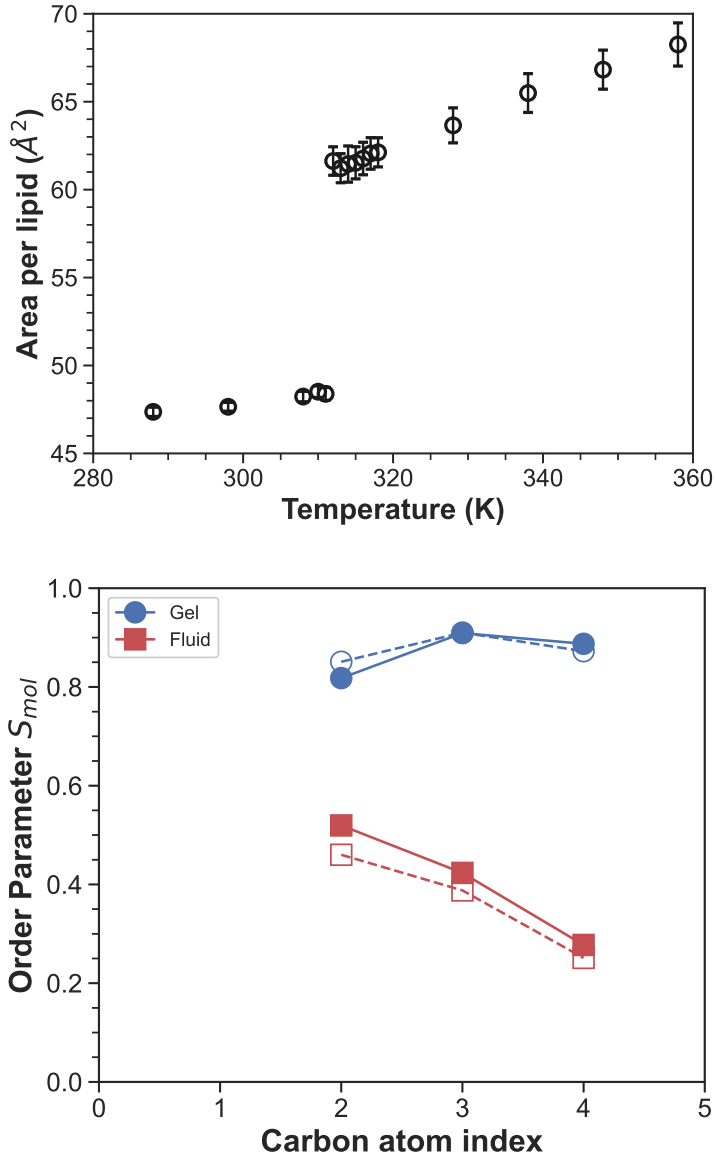


Figure 8: Measurements used to confirm that the DPPC systems, based on the Martini coarse-grain model this time, simulated underwent a phase transition and could be used to train MLLPA. (Top) Evolution of the area per lipid of the system with the temperature, and (Bottom) tail order parameters measured for the system found in gel (288 K) and fluid (358 K). The order parameters have been averaged over the number of lipids and frames in the trajectory. Plain and open symbols are respectively the sn1 and sn2 tails of the lipids.

LIPID	MODEL	SCORES		
		Total	Gel	Fluid
DPPC Charmm	Final	98.0 ± 0.9	98.6 ± 0.8	97 ± 1
	SVM (c.)	97 ± 1	95 ± 3	98.3 ± 0.9
	KNN	95 ± 2	91 ± 3	99.5 ± 0.5
	SVM (d.)	97.6 ± 0.7	97 ± 1	98 ± 1
	NB	96 ± 1	98.6 ± 0.6	93 ± 2
DMPE	Final	89 ± 2	88 ± 3	91 ± 4
	SVM (c.)	88 ± 2	85 ± 3	92 ± 2
	KNN	87 ± 2	81 ± 3	94 ± 2
	SVM (d.)	88 ± 2	88 ± 3	89 ± 3
	NB	87 ± 2	86 ± 2	87 ± 3
DPPE	Final	97 ± 1	97 ± 2	96 ± 1
	SVM (c.)	95.8 ± 0.8	94 ± 2	98 ± 1
	KNN	95 ± 1	91 ± 3	99 ± 2
	SVM (d.)	96 ± 1	96 ± 2	97 ± 2
	NB	94 ± 1	97 ± 1	92 ± 2
DPPC Martini	Final	86.8 ± 0.7	86 ± 3	87 ± 2
	SVM (c.)	87 ± 1	84 ± 3	90 ± 2
	KNN	86 ± 2	81 ± 4	92 ± 2
	SVM (d.)	79 ± 2	75 ± 4	85 ± 3
	NB	77 ± 3	73 ± 4	82 ± 3

Table 2: Complete details of the scores obtained during the training of the different lipid types (DPPC, DMPE and DPPE) and models (all-atom, coarse-grain) used in this paper. The scores were calculated by MLLPA by repeating and averaging the training 10 times with different shuffling of the training dataset and a validation set with 20% of the size of training set.

2.4 Ternary phase diagram

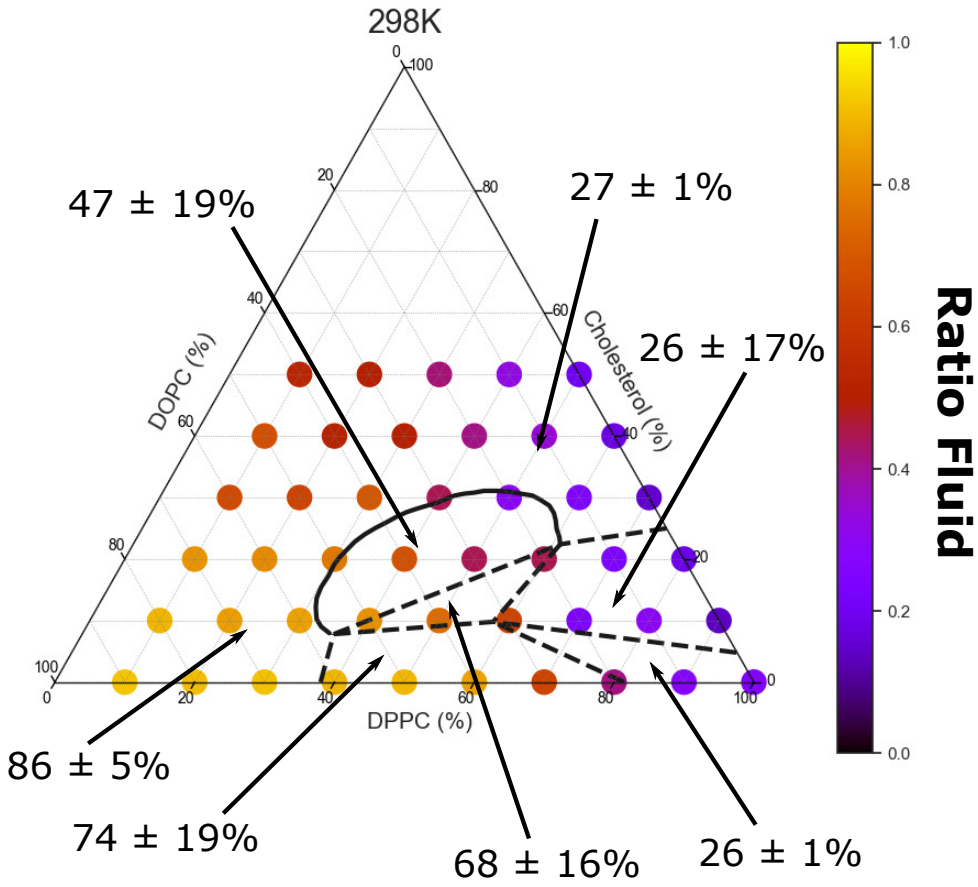


Figure 9: Phase diagram of ternary membranes made of DPPC/DOPC/Cholesterol at different ratios and simulated at 298 K. Each point represents the result of a simulation run. The plain and dashed black lines are the experimental phase diagram as found by Veatch et al.[5] at the same temperature.

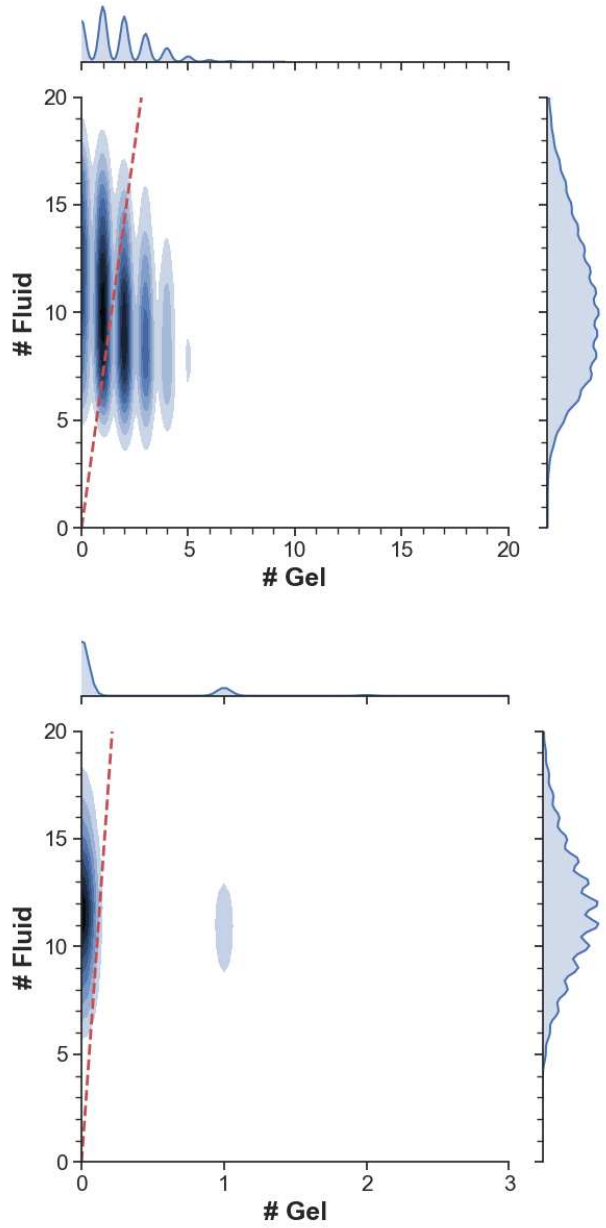


Figure 10: Color coded histograms of the (# gel, # fluid) distribution of probability to find a cholesterol molecule in a environment composed of a given amount of gel and fluid lipids measured in bilayers made of DPPC/DOPC/Cholesterol mixtures of (Top) 50/40/10 and (Bottom) 10/80/10. The red dashed lines shows the measured gel/fluid ratio in all lipids (DPPC + DOPC).

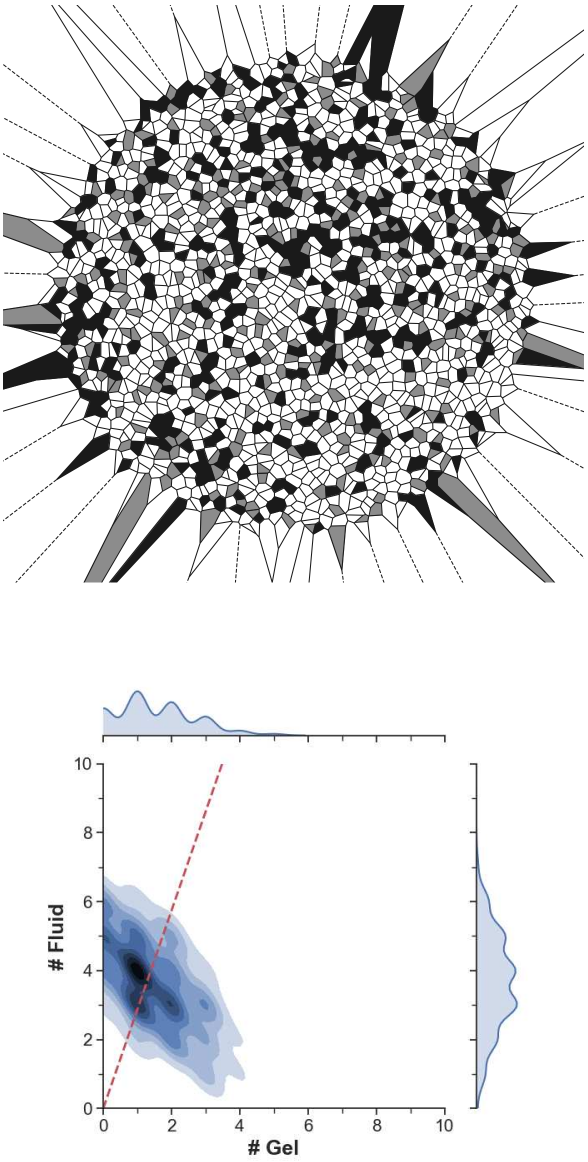


Figure 11: (Top) Voronoi tessellation of the Lambert azimuthal equal-area projection of a 70/5/25 DPPC/DOPC/Cholesterol vesicle simulated using the coarse-grain force field. To gain in readibility, the position of the center of mass of the lipids are not shown in the projection. DPPC lipids found in the gel state are shown in black, while both DPPC and DOPC found in the fluid state are shown in white. This color code was picked in order to facilitate the comparision with the typical experimental representation of phase domains in vesicles[6]. Cholesterol molecules are coloured in gray. (Bottom) Color coded histograms of the (# gel, # fluid) distribution of probability to find a cholesterol molecule in a environment composed of a given amount of gel and fluid lipids measured in the vesicle shown above. The red dashed lines shows the measured gel/fluid ratio in all lipids (DPPC + DOPC).

2.5 Exotic phases

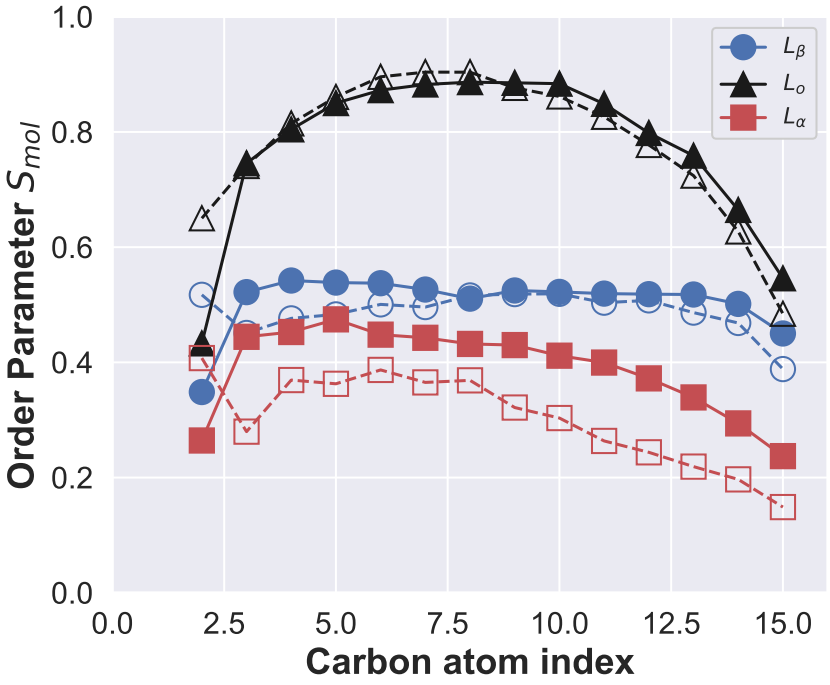


Figure 12: Measurement of the tail order parameters of DPPC molecules in three DPPC/DOPC/Cholesterol systems simulated at 298 K to generate the ternary phase diagram. The respective phases given were determined by comparison with phase diagrams found in the literature[5]. The order parameters have been averaged over the number of lipids and frames in the trajectory. Plain and open symbols are respectively the sn1 and sn2 tails of the lipids.

TRAINING	MODEL	SCORES			
		Total	L_β	L_o	L_α
All phases	Final	98.8 ± 0.4	98 ± 1	98.5 ± 0.9	99.6 ± 0.9
	SVM (c.)	71 ± 2	59 ± 6	65 ± 5	90 ± 3
	KNN	98.9 ± 0.5	98 ± 1	98.6 ± 0.9	100.0 ± 0.0
	SVM (d.)	91 ± 1	92 ± 3	86 ± 2	96 ± 1
	NB	74 ± 2	73 ± 5	68 ± 2	81 ± 3
$L_\beta$vs. L_α	Final	99.7 ± 0.3	99.4 ± 0.7		100.0 ± 0.0
	SVM (c.)	92 ± 2	92 ± 4		92 ± 4
	KNN	99.7 ± 0.3	99.3 ± 0.7		100.0 ± 0.0
	SVM (d.)	96.9 ± 0.9	96 ± 1		98 ± 2
	NB	87 ± 2	87 ± 4		88 ± 4
$L_o$vs. L_α	Final	99.7 ± 0.3		99.7 ± 0.4	99.7 ± 0.5
	SVM (c.)	95 ± 2		92 ± 4	99.5 ± 0.6
	KNN	99.9 ± 0.2		99.7 ± 0.4	100.0 ± 0.0
	SVM (d.)	97 ± 2		89 ± 4	91 ± 2
	NB	90 ± 2		89 ± 4	91 ± 2
$L_\beta$vs. L_o	Final	98.3 ± 1.0	98 ± 2	98 ± 1	
	SVM (c.)	64 ± 2	64 ± 9	67 ± 7	
	KNN	98.4 ± 0.9	98 ± 1	99 ± 1	
	SVM (d.)	89.9 ± 0.8	93 ± 2	87 ± 1	
	NB	76 ± 3	78 ± 3	75 ± 4	

Table 3: Complete details of the scores obtained during the different types of training performed on the three DPPC/DOPC/Cholesterol systems simulated at 298 K and found in the L_β, $L_o$and $L_\alpha$phases (*cf.* Figure 12). The scores were calculated by MLLPA by repeating and averaging the training 10 times with different shuffling of the training dataset and a validation set with 20% of the size of training set.

TRAINING	PHASE	SYSTEMS L_β	L_o	L_α
All phases	L_β	75 ± 8	30 ± 10	10 ± 10
	L_o	22 ± 8	70 ± 10	20 ± 10
	L_α	3 ± 2	2 ± 2	70 ± 20
$L_\beta$vs. L_α	L_β	97 ± 2	94 ± 4	20 ± 20
	L_o			
	L_α	3 ± 2	6 ± 4	80 ± 20
$L_o$vs. L_α	L_β			
	L_o	84 ± 2	97 ± 3	20 ± 10
	L_α	16 ± 3	3 ± 3	90 ± 10
$L_\beta$vs. L_o	L_β	78 ± 8	40 ± 10	70 ± 20
	L_o	22 ± 8	60 ± 10	30 ± 20
	L_α			

Table 4: Ratio of each DPPC phase found by MLLPA in three DPPC/DOPC/Cholesterol systems simulated at 298 K and respectively in the L_β, L_o and L_α phases (*cf.* Figure 12). The scores of four distinct training are compared: a training on the three phases, a training on only the L_β and L_α phases, a training on only the L_o and L_α phases and a training on only the L_β and L_o phases. Cells in pale blue shows the phase ratio found by MLLPA in the same system used to train the model, highlighting potential issues found by MLLPA.

References

[1] V. Walter, C. Ruscher, O. Benzerara, C. M. Marques and F. Thalmann, *Physical chemistry chemical physics : PCCP*, 2020, **22**, 19147–19154.

[2] Y. Qi, H. I. Ingólfsson, X. Cheng, J. Lee, S. J. Marrink and W. Im, *Journal of Chemical Theory and Computation*, 2015, **11**, 4486–4494.

[3] P. Hsu, B. M. H. Bruininks, D. Jefferies, P. C. T. de Souza, J. Lee, D. S. Patel, S. J. Marrink, Y. Qi, S. Khalid and W. Im, *Journal of Computational Chemistry*, 2017, **38**, 2354–2363.

[4] R. W. Hockney, S. P. Goel and J. W. Eastwood, *Journal of Computational Physics*, 1974, **14**, 148–158.

[5] S. L. Veatch, O. Soubias, S. L. Keller, and K. Gawrisch, *PNAS*, 2007, **104**, 17650–17655.

[6] S. L. Veatch and S. L. Keller, *Biophysical Journal*, 2003, **85**, 3074–3083.