Biochimica et Biophysica Acta 1858 (2016) 2468–2482

Contents lists available at ScienceDirect

Biochimica et Biophysica Acta

journal homepage: www.elsevier.com/locate/bbamem

Efficient preparation and analysis of membrane and membrane protein systems☆

Matti Javanainen [a,*,1], Hector Martinez-Seara [a,b,*,1]

[a] Department of Physics, Tampere University of Technology, Tampere, Finland
[b] Institute of Organic Chemistry and Biochemistry, Academy of Sciences of the Czech Republic, Prague, Czech Republic

ARTICLE INFO

Article history:
Received 30 December 2015
Received in revised form 23 February 2016
Accepted 25 February 2016
Available online 4 March 2016

Keywords:
Tools and software
Membrane building
Protein insertion
Molecular dynamics
Lipid bilayer

ABSTRACT

Molecular dynamics (MD) simulations have become a highly important technique to consider lipid membrane systems, and quite often they provide considerable added value to laboratory experiments. Rapid development of both software and hardware has enabled the increase of time and size scales reachable by MD simulations to match those attainable by several accurate experimental techniques. However, until recently, the quality and maturity of software tools available for building membrane models for simulations as well as analyzing the results of these simulations have seriously lagged behind.

Here, we discuss the recent developments of such tools from the end-users' point of view. In particular, we review the software that can be employed to build lipid bilayers and other related structures with or without embedded membrane proteins to be employed in MD simulations. Additionally, we provide a brief critical insight into force fields and MD packages commonly used for membrane and membrane protein simulations. Finally, we list analysis tools that can be used to study the properties of membrane and membrane protein systems. In all these points we comment on the respective compatibility of the covered tools.

We also share our opinion on the current state of the available software. We briefly discuss the most commonly employed tools and platforms on which new software can be built. We conclude the review by providing a few ideas and guidelines on how the development of tools can be further boosted to catch up with the rapid pace at which the field of membrane simulation progresses. This includes improving the compatibility between software tools and promoting the openness of the codes on which these applications rely.

This article is part of a Special Issue entitled: Biosimulations edited by Ilpo Vattulainen and Tomasz Róg.

1. Introduction

The first simulation on soft matter was performed 30 years ago, and since then the field of computational biophysics has expanded at an enormous pace. This first Monte Carlo simulation, studying the water–lipid interface [1], was followed by studies on micelles [2] and bilayers [3,4] using the molecular dynamics (MD) method. Simulations of membrane proteins took place soon after [5,6].

Since these ground-breaking studies in the early and mid 90s, both computing power and the accuracy of the employed models have increased drastically, leading to a large number of studies on membranes (see *e.g.* [7,8]) and membrane protein systems (see *e.g.* [9–11]). What is more, experimental techniques have also improved, providing more accurate data against which the simulation models can be parameterized and optimized. Nowadays the knowledge required to perform MD simulations of membranes or membrane protein systems is easily available for everyone *via* the internet. Such simulations can be performed with numerous available software packages, including several free reliable alternatives, on any modern desktop computer to a certain extent.

However, except for the last few years, what has been seriously lacking are publicly available user-friendly tools that aid the setting up and analysis of membrane or membrane protein simulations. Such tools are necessary to make the field of computational biophysics more approachable to newcomers. Additionally, they would also simplify the tasks of experienced scientists, as automation and ease-of-use of tools would leave more time for the actual science. Luckily things are changing and a number of new approaches have been introduced to both building lipid membranes and inserting proteins into them, as well as to analyzing the results of the simulations on these systems.

Most of these new tools have been made available since the last thorough review on the topic almost ten years ago [12], which calls for an update. In this paper we review the important aspects of setting up and analyzing membrane and membrane protein simulations. It should be noted that this review does not aim to provide step-by-step instructions for performing membrane or membrane protein simulations, yet such recipes are available in *e.g.* Refs. [13] and [14]. Instead,

☆ This article is part of a Special Issue entitled: Biosimulations edited by Ilpo Vattulainen and Tomasz Róg.

* Corresponding authors.
E-mail addresses: matti.javanainen@tut.fi (M. Javanainen), hseara@gmail.com (H. Martinez-Seara).

[1] Both authors contributed equally to the article.

http://dx.doi.org/10.1016/j.bbamem.2016.02.036
0005-2736/

we aim to provide a comprehensive list of the key software available. We comment the ease-of-use and generality of these tools and also provide information on their compatibility with force fields and file formats. This listing will aid both newcomers to select the proper tools for their project as well as inform more experienced users of newly published tools and techniques. It must be noted that the ever increasing user-friendliness of the applications and simulation software might, however, introduce a new and a perhaps surprising issue. Newcomers without the proper background knowledge on the underlying algorithms might nowadays be able to perform both simulations and analyses. This might accidentally lead to incorrect conclusions that are extremely hard to catch during the peer-review process. Therefore, it is important that regardless of how easy to use scientific tools become, they should never be used as black boxes.

This review is structured as follows. We first introduce the most common force fields employed in molecular dynamics simulations of lipids and proteins. Next, the numerous approaches used to build lipid bilayers are reviewed. This is followed by a thorough list of techniques and tools for the insertion of proteins into membranes. After a brief examination of the popular molecular dynamics software packages, we review tools available for the analysis of membrane and membrane protein simulations. Finally, we raise issues related to the current paradigm of tool development and try to foresee how these issues could be tackled in the near future.

2. Force fields for biomolecular simulations

A careful selection of the proper lipid and protein force fields is of key importance for every project considering MD simulations on biomolecular systems. Most importantly, the level of detail of the chosen force field, be it *e.g.* a fully atomistic or a coarse-grained one, should allow to sample time and size scales relevant for the problem at hand yet still provide the required chemical accuracy. Another factor affecting the selection of the force field is its compatibility with the available simulation software. What is more, the chosen force fields should either include the molecule parameters related to the research problem or provide tools for parameterizing them. Lipid force fields seldom cover all possible head groups and tail types. Notably, certain head groups (such as phosphatidylcholine) and tails (such as palmitic acid or oleic acid) are often parameterized first and appear in almost every lipid force field. On the other hand, some head groups (such as phosphatidylinositol) or tails (such as linoleyl or linolenoyl) are rarely available. Therefore, the desired membrane composition might limit the number of plausible force fields. The choice of the lipid force field also often sets limits to the available options for the protein force field, and *vice versa*. Sometimes the projects involve molecules beyond lipid and protein families (such as sugars or nucleotides) and in such cases the selected force field should also cover these extra molecule types or be compatible with a force field that contains them.

Some common force field models, which can be divided into different categories based on how much detail they provide, are briefly listed below. For more thorough reviews and comparisons of lipid and protein force fields please see Refs. [15–19]. Notably, no thorough comparison of the performance of the force fields in describing membrane protein systems exists in the literature to our knowledge.

2.1. Coarse-grained models

Coarse-grained models map multiple atoms into larger pseudoatoms or "beads", which significantly reduces the number of degrees of freedom and therefore allows longer simulation times.

The Martini model has gained broad acceptance in the biomolecular simulation community. It contains parameters for lipids [20], including glycolipids [21], and proteins [22,23] as well as carbohydrates [24] and nucleic acids [25] among others. It is also compatible with a polarizable water model [26]. The implicit solvent version of the Martini lipid force field, titled Dry Martini, is also available [27]. One major advantage of Martini, in addition to the large selection of parameterized molecule types, is the number and quality of tools provided on the Martini website.

The PLUM force field also relies on a solvent-free approach and contains parameters for both proteins and lipids [28–30]. One key advantage that PLUM has over Martini is that it describes protein folding, whereas secondary structures are fixed in Martini.

Furthermore, the ELBA force field [31] introduces dipoles into both lipid molecules and water beads, which greatly improves the description of electrostatics. However, the number of lipid types available is very limited and proteins have not been parameterized at the time of writing this review.

2.2. United-atom force fields

United atom models usually combine methyl groups and methylene bridges into pseudoatoms, thus effectively combining the properties of the hydrogen atoms into their host carbons. The most common of such force fields, namely GROMOS, contains multiple parameter sets for proteins with the newest one being 54A7/54B7 [32]. The multiple versions are also compatible with the corresponding lipid force fields [33–35] and contain parameters for many other molecule types, such as carbohydrates and nucleic acids. Two automated web-based tools exist for the parameterization of small molecules for GROMOS. The long-running and popular PRODRG server [36,37] has recently received criticism, most importantly for its poor handling of charge groups [38]. The more recent Automated Topology Builder (ATB) [39,40] aims to tackle the charge group partitioning issue [41], in addition to other improvements.

Here, the commonly employed yet old Berger united atom lipid model [42] should be mentioned. It combines parameters from multiple sources and has been used together with atomistic protein force fields (see below). This parameterization was recently refined to correctly describe phase behavior [43].

The united atom TraPPE force field contains parameters for lipids [44] yet parameters for proteins are not available.

2.3. Atomistic force fields

Thanks to the rise in computing power, researchers can now waive the performance provided by united atom approaches in favor to the improved accuracy provided by fully atomistic force fields. Additionally, the interest towards membrane protein simulations has called for the development of high quality lipid force fields compatible with the protein force fields previously employed in simulations of water-soluble proteins.

Various versions of the Amber protein force field are commonly used, with ff99SB-ILDN [45] gaining widespread acceptance. Additionally, the ff99SB force field was recently refined in the form of ff14SB [46]. Further, another development branch entitled ff14ipq employed charges derived in a new way [47] and has not yet been thoroughly tested. Even the old ff03 is still used to some extend [48] (note that ff03 is from 2003 whereas ff99SB-ILDN is from 2010).

Multiple Amber-compatible sets of lipid parameters also exist. The General Amber Force Field (GAFF) lipid parameters [49] were later combined with the development of Lipid11 [50] resulting in the Lipid14 parameter set [51,52]. Lipid14 contains parameters for several lipid types as well as cholesterol. Until Lipid14 all these force fields required the use of applied surface tension in order to maintain the membrane in a liquid phase. In addition, the Slipids parameter set [53–55] is compatible with the Amber protein force fields and has parameters for multiple lipid types including sphingomyelin and cholesterol. However, polyunsaturated tails are not included in Slipids. Amber also supports the Glycam carbohydrate force field [56]. Automated ways to parameterize molecules, such as drugs, for the GAFF [57] force

field are available [58–61]. Additionally, a database containing a number of parameterized molecules for GAFF (as well as OPLS-AA and CHARMM general force field) in the GROMACS format is available online [62].

The CHARMM22 protein force field [63] together with the CMAP correction [64] as well as the more recent CHARMM36 protein force field [65] have been employed extensively to study membrane proteins. The recently launched CHARMM36 lipid force field also allows the simulation lipid bilayers in a tensionless state. This CHARMM36 lipid parameter set [66] includes a great number of the most common cell lipid types [67]. CHARMM also includes parameters for carbohydrates [68] and nucleic acids [69]. A MATCH server providing automated atom typing [70] exists for CHARMM and the CHARMM general force field (CGenFF) [71] also supports ways to automate the force field parameter generation [72,73,59]. A large set of parameterized molecules for the CGenFF is also available in the GROMACS format [62].

The OPLS-AA all atom protein force field [74] and the fresh update (OPLS-AA/M) [75] are compatible with the recently released lipid force field [76,77]. Unfortunately, a very limited set of lipids is currently available. OPLS-AA is also compatible with a limited set of carbohydrates [78], and a large set of parameterized molecules is available for OPLS-AA in the GROMACS format [62].

2.4. Hybrid approaches

To combine the speed and detail benefits of models at two levels of resolution, multiple hybrid approaches have been developed. The united atom Berger force field [42], introduced earlier, has been combined with both Amber [79] and OPLS-AA [80] protein force fields.

The PACE force field was designed to combine a coarse-grained membrane model with a united atom protein model [81–83].

Similarly, the Multi-Scale Coarse-Grained (MS-CG) force field has been used to combine a lower-detail membrane into a higher-detail protein [84]. Similarly, the Martini force field has also been combined with higher-detail proteins [85].

2.5. Polarizable force fields

Polarizable force fields, as their name suggests, aim to account for the polarization of the media due to the presence of charged molecules. To our knowledge, only the polarizable CHARMM force field based on the Drude oscillator contains parameters for both lipids [86] and proteins [87]. Additionally, nucleic acids have been parameterized for this force field [88]. These models, however, have not been widely adapted due to their computational cost and fairly limited accuracy. More information on them can be found in a number of very recent reviews [89–91].

2.6. Transformations between models

Tools allowing flexible transformations between different models have been developed recently. They allow the fine-graining of coarse-grained models into either the fully atomistic or the united atom scheme. To our knowledge three independent such tools exist [92–94], each with their own features and limitations. Additionally, Lipid Converter [95] allows conversions between lipid force fields (all-atom and united atom) as well as between lipid types within one force field.

3. Building membranes

Biological membranes are very complex entities. They are constituted by a vast number of different lipids, proteins and sugar moieties. The proportions of these moieties and their spatial distribution depend largely on the studied membrane [96] and may even change in time. With such complexity, it is understandable that one of the critical tasks related to membrane simulations is the construction of the system that is to be simulated.

So far, molecular modeling of membranes has focused on simulating small bilayer patches containing just few lipid moieties. Although simplified, these systems are believed to be representative models for some aspects of real biological membranes. Within this framework, the construction of membranes was usually performed manually and each researcher had his or her own protocol for this task. As long as the membranes were simple enough and the number of systems required for a project remained rather small, this approach was tolerable. However, this was a clear impediment for newcomers who wanted to include membrane simulations in their research toolkit. What is more, even experienced membrane researchers had to spend substantial amount of time on this task rather than on actual science.

This has drastically changed during the last few years. Several new tools and approaches that aim to simplify and avoid the pitfalls of previously established protocols have appeared. Coupled with the increase of computing power, this has changed radically the complexity and size of studied membranes. It is nowadays not rare to see simulations of large membranes including many lipid moieties in combination with other molecular classes like proteins [97–99] and even polymers [100]. Such systems would be too complex to be set up using manual membrane building protocols.

This section aims to review the currently available methods for constructing lipid-only membranes with emphasis on their known strengths and weaknesses. The construction of protein-containing systems as well as insertion of proteins into membranes will be discussed in the next section.

We will start by introducing a prototype of the manual building protocol. This should serve to understand why the other, more straightforward methods introduced later in this section, are needed. We will also point out the existence of several repositories where one can find specific pre-built and even equilibrated membranes ready to be simulated. We will then present several tools which allow the construction of custom membranes. Finally, we will briefly discuss the improvements expected from future approaches.

3.1. Previous considerations

A few important points need to be considered before selecting the method for building a membrane for a particular project. The first step is to choose the membrane composition. Decision of the required lipid moieties together with their desired proportions should be based on experimental evidence of the systems that are to be mimicked [96]. However, this information is not always fully available since the extraction of specific membranes and the subsequent separation of lipid components are challenging experimental tasks. What is more, there are also technical aspects that need to be considered. Most importantly, not all possible lipids, *i.e.* naturally occurring or artificially synthesized, have been parameterized. This is a fundamental point because parameterizing lipids is a time consuming and demanding task. Therefore, unless one is willing to devote the time required to parameterize the considered lipid, researchers usually stick to those currently available even if this results in small deviations from the desired optimal composition. Moreover, not all available lipids have been parameterized for all force fields. Therefore, to avoid large compromises in the desired membrane composition, the choice of the force field is often limited to few options, and this choice also dictates the set of tools that can be used to build and simulate the membrane, as discussed below.

Several approaches aim to overcome this problem, yet they have to be considered with extreme caution. There are tools that, despite never meant to be used to parameterize lipids, can be used to perform this task [49]. Another option is to adapt the missing parameters from a different force field. It is also common to build topologies for new lipids by complementing existing ones using chemical analogy of known molecular fragments [101]. Although the agility of these shortcuts

might look attractive at first, they should be avoided. They rarely work properly and even if they do, the quality does not likely match that of the carefully parameterized models. Therefore, extensive validation of the behavior of the produced lipids must be performed in order to avoid artifacts.

3.2. Structure and topology of lipids

In comparison with other biomolecules, naturally occurring lipids are relatively small and present simple connectivity [96]. One can construct them easily using any molecular editor, *e.g.* Avogadro [102], in combination with any chemical software toolbox, *e.g.* Open Babel [103], that provides connectivity information required for the molecular topology. As lipids contain stereochemical centers, which are often exclusively selected in nature, one must ensure that the constructed lipids match the required spatial configuration [96]. Despite the apparent simplicity of this process, the real challenge lies in finding suitable force field parameters for the constructed lipid. Lipids are known to be very challenging molecules to parameterize [104], as high-level quantum chemistry calculations are required in order to obtain values for the partial charges and dihedral potentials. Furthermore, the Lennard-Jones interactions need to be iteratively adjusted against available experimental data. In general, this iterative nature of the parameterization process restricts the use of the lipid force field to a very particular water model and to a very narrow set of parameters, such as the cut-off scheme of nonbonded interactions [104]. Furthermore, when combined with other molecule types, the parameters should be consistent, which in general means that the same protocol needs to be followed in the parameterization of all associated molecules. This restriction renders the shortcut approaches described above fairly useless, unless the changes to existing lipid models are cosmetic, *e.g.* extension of acyl chains. In any other case one should download the required structures and topology files from one of the several available web resources, such as lipidbook [105], CHARMM-GUI [106], as well as Slipids [55] and Martini [20] websites. In some rare cases, force fields can also be provided by the used MD package [51].

Recently, taking advantage of the modular design of the CHARMM force field, a new tool called LipidBuilder [101] has been developed. This tool can be used to generate certain customized lipid topologies for this force field by combining existing head groups into tails of desired length and level of saturation. Although, its use is rather limited to NAMD MD package and just one force field, it points to the correct direction by solving the problem of lipid structure and topology generation.

3.3. Manual construction of membranes

Building a membrane manually starting from the known structure of the desired lipids is usually a tedious, highly inefficient and even error-prone task, even in the case of simple single-component lipid bilayers [13]. All manual protocols share essentially the same steps, as shown in Ref. [13].

The process starts by spreading oriented lipid molecules in a grid. The coordinates of one lipid are replicated and translated to match the location of the assigned grid points. For this reason, the positions of the lipids in the resulting bilayer are usually highly correlated. This is followed by solvation and a subsequent long equilibration phase. To avoid collisions between lipids and penetration of ring structures, the initial distances between lipids need to be fairly large. This results in a long and tricky equilibration simulation, during which the bilayer shrinks until it relaxes to the correct area per lipid. In order to speed up this process, the equilibration can be performed without water molecules by restricting the movement of some lipid atoms in the direction normal to the membrane [107]. This trick also prevents lipid flip-flops, which are fairly common within this loosely packed initial structure. What is more, this approach also minimizes the amount of water that ends up inside the membrane during the posterior solvation process since the excess free volume within the membrane into which water molecules could be accidentally placed has already been eliminated.

Another important problem arises with multicomponent membranes. In these systems the components should be carefully mixed from the very beginning since the lateral diffusion of lipids is slow [108]. Even for small membrane patches in a fluid phase the real mixing process might take close to a microsecond [109]. The mixing time increases very steeply with the number of lipids. Therefore, for large membranes, the initial lipid positions need to be properly randomized. Otherwise, microseconds of simulation might be wasted just for equilibration and lipid mixing.

Finally, on many occasions several initial structures of the same membrane are generated and simulated to serve as independent replicas of the studied system. This approach aims to tackle the limited sampling problem that is expected in the currently reachable time scales. Unfortunately, when manual methods are used to generate membranes, the construction of such independent configurations essentially requires repetitive manual work.

All these problems clearly call for other membrane building methods that are free of the listed shortcomings. Such tools will certainly not only improve the efficiency of the research workflow, but also prevent several typical and difficult-to-spot mistakes.

3.4. Pre-equilibrated bilayers

One of the first alternatives to manual construction of membranes was the reuse of pre-equilibrated structures. Alternatively, larger membranes could be constructed by simple multiplication of the coordinates of these structures. One advantage of reusing membranes is that their integrity is often better cross-checked by other researchers. This reduces the chance of errors and allows direct comparison with previously published data obtained with the same membrane model. It is also possible to slightly change the lipid composition of pre-equilibrated membranes by selectively removing some lipids. Further, modifying lipid moieties to other ones is possible [95] and also very efficient when coarse-grained models are employed. However, such modifications must be performed very carefully to atomistic systems in order to avoid affecting the structural connectivity of the lipids such as their stereochemistry.

Overall, these arguments convinced several research groups that have been publishing the membrane structures used in their research on their group websites. Despite being appreciated initiatives, these collections provide a solution that is far from optimal as the membrane structures are scattered across numerous hard-to-find websites. New protein structures are stored without exceptions in the centralized RCSB PDB repository [110] and a few initiatives such as lipidbook [105] or more general repositories (*e.g.* www.zenodo.org) might steer also the membrane simulation community to this direction in the near future.

Unfortunately, using pre-existing membranes has several drawbacks. To begin with, this approach is not flexible at all. Furthermore, the ordering of lipid atoms likely depends on the employed force field. Therefore, either the same force field with which the membrane was originally simulated needs to be used, or the atoms need to be reordered. Despite being a fairly simple task, the reordering process is rather prone to errors that can lead to subtle and hard-to-spot changes in *e.g.* chirality of stereocenters and cis–trans isomerization.

3.5. Membrane builder applications

To solve the issues mentioned above, especially those related to the building of custom membranes, new tools have recently started to emerge. These tools usually provide a fully automated mechanism to distribute different lipid moieties on top of a grid, with the exception of CHARMM-GUI [111] that provides membrane structures with lateral

density already close to its equilibrium value. In all the tools mentioned below lipids are randomly distributed to improve their lateral mixing.

The tools can be divided into two categories: web services (CHARMM-GUI [111], Membuilder [112], LipidBuilder [101]) and distributed applications (Packmol [113] and insane [114]). Web-based applications usually provide a user-friendly interface, and they often generate not only the coordinate files but also all other files required to perform simulations on the system, *i.e.* topologies and simulation parameters. Their downside is that their performance is often low or even limited by design to optimize the web server resources. Generating highly optimized membranes is a resource-intensive process. Taking into consideration the limited resources that a research group can divert to such a service, these problems might prove difficult to address. Another aspect of such web-based tools that restricts their usage in innovative research is that only few people (*i.e.* the developers) can modify them. Only if all the requirements of a project are fulfilled by a web service, can full advantage be taken out of it. This is true even when trivial modifications, such as the addition of a new lipid type, are required. In fact, this is a common problem of all these services. They can only be used for a limited number of MD packages, lipid moieties and force fields.

The newly published LipidBuilder [101] partly addresses the limitation of fixed set of available lipid moieties by including a lipid builder application, yet it is only compatible with the CHARMM force field and the NAMD software. The limitations of another web service, Membuilder (with Membuilder II available as a beta version at the time of writing this review), are related to MD software, as it only supplies files in the formats of GROMACS. However, a number of GROMOS (43a1,43A1-S3, and 53a6) and CHARMM (27 and 36) as well as the Slipids force fields are supported.

CHARMM-GUI [115] nowadays provides input files for multiple MD packages including GROMACS, NAMD, Amber, OpenMM and CHARMM, yet in terms of atomistic force fields it is limited to CHARMM36 [66]. Naturally, however, the structure provided by CHARMM-GUI can also be used with other force fields by simply altering the atom ordering if it differs from that of CHARMM36. Finally, it should be mentioned that these web services are difficult to link to other tools, which further limits their usability and adoption as a *de facto* tool.

On the other hand, distributed software tools for membrane generation are still in their infancy. With the exception of insane [114] that generates sophisticated membranes for the Martini force field [20], the features provided by membrane builder software fall short compared to those of their web-based counterparts. The only program really worth mentioning is Packmol [113] which allows the efficient generation of any kind of densely packed structures, including sophisticated membranes. However, this software is blind to molecular details and considers molecules as rigid entities. Therefore, it cannot obtain similar optimal packing densities as CHARMM-GUI.

3.6. Building micelles, liposomes and other non-planar structures

Besides planar membranes, other lipid structures bear biological interest. Micelles, vesicles, and hexagonal phases are notable examples of such structures. Some tools, such as CHARMM-GUI [111], Membuilder [112], and Packmol [113] can also create such structures within their limitations.

3.7. Final remarks

To conclude, new tools that simplify the creation and simulation of complex membranes are constantly being developed. These tools represent a significant improvement with respect to the previous paradigm where manual setup or reuse of existing membrane structures were the only available options. Taking into consideration the number of tools that appeared recently and their constant improvement, we can expect that running a membrane simulation will become a rather simple task in the near future. However, building tools are useless if the used force field does not provide parameters for the desired lipids. So far, an automated way to reliably parameterize lipids does not exist and its implementation seems like a monumental task. However, development in this direction would pave the way for the ultimate general membrane builder tool which would be free of all the numerous limitations listed above.

4. Insertion of membrane proteins

As membrane protein simulations have recently become increasingly popular, the quality and variety of tools for constructing the initial structures for such simulations have also improved substantially. Notably, most of the tools and approaches reviewed in this section were developed less than five years ago, which calls for an update to the thorough review of Kandt et al. [12].

Having set up a lipid membrane following one of the approaches listed in the previous section, a choice between the multiple protein embedding approaches needs to be taken. Alternatively, the protein containing membrane can also be assembled from scratch. As in the case of building a membrane, none of the possible approaches can be declared to be the best in every possible case. Instead, a choice needs to be made based on multiple criteria. The approach must be compatible with the chosen lipid and protein models. Some approaches work with multiple force fields, even covering both atomistic and coarse-grained schemes, yet some were designed to work strictly with a single force field. Additionally, the file formats involved in the process need to be compatible with the ones at hand, or at least a straightforward conversion must be available.

The protein embedding approaches can be divided into three categories. First, web-based tools provide a simple way to conduct the task, yet they are often not adaptable to cases other than the ones for which they were originally designed. Second category includes methods that employ the functionality of molecular dynamics packages or are included exclusively within a certain MD code. Third, some approaches are based on independent software designed solely for the purpose of protein embedding. It must also be mentioned that, like with membrane structures, some websites offer pre-equilibrated membrane structures with membrane proteins embedded into them. Naturally, these available structures seldom provide the exact desired composition and, as mentioned in the previous chapter, modifications to their composition must be performed with care.

In this section we list the existing approaches for protein insertion and evaluate them in terms of ease-of-use, computational cost and, most importantly, compatibility in terms of both simulation software and the employed lipid and protein models.

4.1. Early approaches

The first simulations of membrane protein systems [5,6] relied on assembling a membrane around a protein. With proper automation and with a sufficient collection of lipid conformations from which the membrane is built, this approach can be quite effective. However, since the packing algorithms are never perfect, the built system might require extended energy minimization and equilibration simulations with restraints on the protein structure. In case molecules with rings, such as sterols, are involved, care must be taken to avoid ring penetration by *e.g.* lipid tails. Also, no advantage of a pre-built and equilibrated bilayer can be taken. It is noteworthy that the CHARMM-GUI membrane builder, discussed later in this section, implements this approach. Another and more general tool which can be employed to build the bilayer around a membrane is Packmol [113], a software which places molecules in a system based on user-provided restrictions. Examples of protein containing systems generated with Packmol and CHARMM-GUI can be found in Fig. 1.

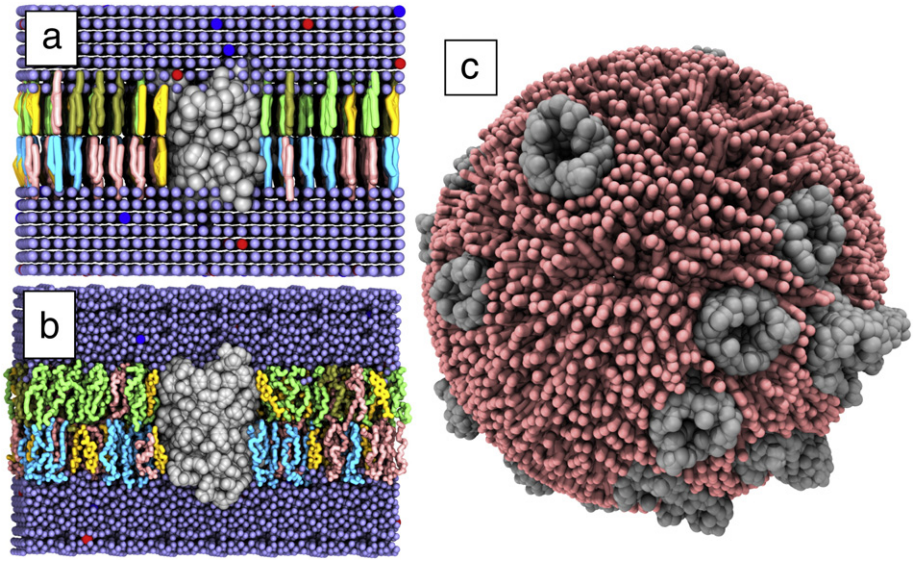

Fig. 1. Outputs of several highly automated tools that can be used to build systems for membrane protein simulations. a) The coarse-grained structure generated by the insane tool. Protein is shown in gray spheres, whereas the different lipid moieties are colored (pink, cyan, yellow, red, tan) to emphasize the tool's ability to build asymmetric membranes containing numerous lipid types. Additionally, water (iceblue) and ions (red and blue) are also added. The initial structure is built on a regular grid from very ordered lipid structures, yet the relaxation during a simulation is very rapid thanks to the coarse-grained scheme. b) An atomistic bilayer generated by the CHARMM-GUI membrane builder. Coloring as on previous panel. c) Multiple proteins (gray) embedded in a DPPC lipid vesicle (pink) in the coarse-grained scheme. The system is built with the Packmol tool. The shown structure was kindly provided by Petri Kaurola.

Another straightforward approach is to delete lipids from a bilayer, creating a void into which a protein can be placed. The removal can be performed manually, or tools such as genbox/solvate of the GROMACS package can be employed to remove lipids which collide with the protein. This simple lipid removal perturbs the lipid composition and the (a)symmetry of the membrane. A gentler approach is described by Shen et al. [116], who first delete a few lipids to create an initial hole, which is then expanded by a cylindrical repulsive potential. The resulting hole does not often match the shape of the protein, which again indicates that a long equilibration is required. This method was developed further by taking into account the protein shape in creating the hole. Approaches following this idea, namely make_hole and GRIFFIN are discussed later in this section.

The clear advantage of these two methodologies is their universality to both atomistic and coarse-grained models as well as independency from the used MD package. However, these approaches are not very user-friendly, they require removal of lipids, and the generated structures need a more careful equilibration than many of the more recent approaches described in the following sections.

4.2. External software and modified MD code

Instead of a simple cylinder, the mdrun_hole (or make_hole) tool [117] creates a hole based on the shape of the inserted protein. This minimizes the empty space around the protein following its insertion, which results in a faster equilibration of the system. This approach, even though employed extensively in the past, comes short due to a few reasons. First, two external pieces of software are required to create the input files for the tool itself. Second, the mdrun_hole relies on GROMACS code and the most recent version is based on an outdated version 3.1.4. This dependency on GROMACS also indicates that all involved files will need to be available in formats that are compatible with GROMACS. Additionally, like its counterpart that creates cylindrical holes mentioned earlier, this approach also requires the removal of lipids from the system, which will perturb the lipid composition and (a)symmetry of the membrane.

The InflateGRO method [12], later updated to InflateGRO2 [118], is also based on creating a hole in the host membrane and placing the protein into this hole. However, the perturbation of the membrane structure is minimized by first inflating the membrane, *i.e.* scaling up its coordinates to create more space between lipids. The protein is then inserted and only few lipids need to be removed due to the large inter-lipid spacing. Finally, the coordinates are stepwise scaled back to the original lipid density and the structure is energy-minimized at each step. Alternatively, in case no pre-built bilayer is available, it can be built with a large inter-lipid spacing and the procedure followed from there on. This approach also requires the removal of lipids, though the number is smaller than in many other methods. Most importantly, the tool relies on the GROMACS engine to perform the energy minimization steps. Therefore, the force field files must be available in a GROMACS-compatible format. Additionally, the scaling script works on GROMACS-based ".gro" file format. However, many of these limitations can be overcome by tools which allow conversion of MD software specific files from one format to another.

Despite its success, the original InflateGRO had a few issues as listed by Schmidt and Kandt [118]. First, the termination of the script after the coordinate scaling was not well defined in many cases. Second, the tool was only functional with atomistic systems. Third, the functionality was based on residue names leading to many complications with heterogeneous lipid membranes. Fourth, the inflation of the membrane could result in loss of equilibrated lipid conformations. Finally, the vertical position of the protein in the membrane had to be manually adjusted. These issues were removed with the update to InflateGRO2 [118]. Bundled with the new aligning tool LAMBADA, the procedure is now fully automated, applicable to coarse-grained systems, and functions based on index files instead of residue names. However, the dependency on GROMACS and the requirement for lipid deletion still remain after the update. Furthermore, whereas the original InflateGRO provided a way to insert lipids into the enclosed volume of donut-shaped proteins, such functionality is not present in InflateGRO2.

Another GROMACS-based approach, g_membed [119] (now a part of the mdrun functionality), is currently included in the GROMACS package. This tool is based on a clever approach to first embed a shrunken protein into a hole created by the removal of a few lipids. The protein is then scaled up slowly to its full size during a short simulation, which causes the membrane to gently adapt to the presence of the protein. Due to this smooth adaptation, the system should only require a fairly short equilibration after the insertion process. Since the

functionality is only available as a part of the GROMACS package, files need to be in a format supported by it. However, the tool is not restricted to any specific force fields.

The idea behind the mdrun_hole methodology (see above) was refined in the GRIFFIN tool [120] that brought the whole procedure into one stand-alone tool and simultaneously advanced its functionality. In addition to the shape-dependent forces employed in the earlier tool to push away lipids and solvent molecules, GRIFFIN also includes Coulombic and van der Waals forces in order to better optimize the lipid–protein interface, and therefore minimize the equilibration period required after the insertion of the protein. Additional geometrical restraints can also be provided for GRIFFIN in order to keep molecules away from undesired locations such as prevent lipids from entering protein pores. GRIFFIN works by providing the MD code with forces that are used to push away lipid and solvent molecules. One important strength of the tool is that it works also with donut-shaped proteins, *i.e.* proteins which encapsulate lipids in some hollow region within their structure. However, as with many methods, GRIFFIN also requires the removal of lipids in order to obtain the correct system density before applying the forces that make room for the protein. Additionally, the tool is limited to work together with GROMACS and NAMD packages.

In the case of coarse-grained models, system equilibration is usually very rapid, and therefore the initial structure does not need to be so carefully constructed. A Python-based tool insane [114] builds up membrane protein systems compatible with the Martini force field. A coarse-grained structure of the protein needs to be provided and it can be easily generated with the martinize tool. The output structure contains all lipids in a straight-tail conformation (see Fig. 1), which is not a problem due to the rapid equilibration. Additionally, no care needs to be put on preserving the secondary structure of the protein since this is fixed in the Martini protein force field. The insane tool can build membranes with complex lipid compositions and it can be employed to insert multiple proteins by exploiting the functionality provided by the DAFT approach [121]. In addition to being limited to only the Martini force field, insane also suffers from the lack of possibility to specify the exact number of lipid molecules of each type as only ratios of lipid components can be provided. This might result in the need to manually remove molecules from the system to reach the desired numbers of the lipid molecules. Some functionality remains to be included at the time of writing this review. One ultimate benefit of this tool is that large complex lipid systems can be easily constructed and rapidly equilibrated in the coarse-grained scheme followed by the transformation into a model providing atomistic detail *via e.g.* the backward tool [93].

To summarize, multiple protein insertion methodologies rely on either external software or are directly built-in in the MD software. These tools are often fairly straightforward to use, yet the dependency on the MD software or related file formats limits their general adaptation. The next section discusses two tools which provide a more universal approach.

4.3. Methods employing the functionality of the MD code

Some recent protein insertion approaches rely on the model-independent functionality available in all common MD software, which renders them essentially universal. In the approach described by Javanainen [107], a protein is first placed next to a bilayer and then inserted into it by applying a high lateral pressure on the system. This approach does not depend on the MD software nor the used lipid and protein models. One additional strength of this method is that there is no need for lipid removal. This means that the lipid composition and (a)symmetry of the host bilayer are maintained throughout the process. However, this also means that in the case of proteins that occupy different volumes in the two membrane leaflets, the number of lipids in these leaflets need to be manually adjusted in order to create a planar stress-free system. Since the method is based on running a simulation on an unmodified MD code using specific pressure coupling options and position restraints, some manual work is required to set up these simulations. The approach also allows the insertion of multiple proteins at once even though this is somewhat complicated. The method is limited to planar geometries and cannot handle donut-shaped proteins.

Another approach, alchembed [122], relying on the built-in functionalities of MD codes pushes the lipids away from the protein volume by slowly turning on the protein's interactions with lipids and other surrounding molecules. This is achieved by soft-core interactions originally developed for free energy calculations. This functionality is therefore readily available in all major MD packages. The method is also general in terms of lipid and protein models, and both atomistic and coarse-grained models can be employed. Similar to the method based on high lateral pressure described above, alchembed also relies on manually positioning and aligning the protein prior to running the simulation during which the interactions are turned on. As a downside, lipids have to be removed manually from the volume into which the protein is inserted. Moreover, parameters to be employed for the soft-core potential are suggested, yet their generality remains to be carefully tested. However, multiple proteins can be inserted at once and non-planar geometries can be employed, even though the manual position and alignment procedures might become quite tedious in such systems.

As described above, the approaches relying on the functionality of the MD packages can be used with essentially all MD software and with all lipid and protein force fields. However, this universality comes with a price: the procedures are not automated and lack the ease-of-use of some other approaches. Quite a lot of manual input is required yet for an experienced user this requirement should be balanced by the versatility of the approaches.

4.4. Web-based tools and data banks

For quite some time many research groups have shared both lipid and lipid–protein structures on their web pages. Even though such systems seldom directly match the requirements of other researchers, they can act as a good starting point from which a desired structure is obtained, for example, *via* changing lipid types, force fields or even level of detail of the model (from atomistic to coarse-grained and *vice versa*).

A significant improvement to this recently surfaced, as the group of Prof. Mark Sansom initiated a thorough data bank [123] containing the structures of all known membrane protein structures embedded in a DPPC bilayer. These pre-equilibrated structures are available in coarse-grained (Martini) and united atom (GROMOS) schemes. Via specialized tools, such as backward [93], the coarse-grained structure can be turned into an atomistic one corresponding to any force field provided that the mapping between them is available or generated. The limitation to a DPPC bilayer is not a major one, since in the coarse-grained scheme transformations between lipid types can be easily performed *via* simple scripts and the equilibration following such modifications is rapid.

Another huge step forward in building membrane protein systems was the introduction and subsequent development of the CHARMM-GUI website [124,125]. CHARMM-GUI enables the flexible construction of various kinds of systems including lipid structures of various geometries as well as membrane protein systems. The lipid composition as well as the type and amount of the solvent can be carefully adjusted. Output files containing the assembled and nearly equilibrated system are provided in various formats compatible with CHARMM, NAMD, GROMACS, AMBER and OpenMM [115]. One downside of CHARMM-GUI is that only membranes containing a single protein can be built. Additionally, in the atomistic scheme only the CHARMM force field is supported. However, with tools such as the Lipid Converter [95] the transformation of the lipid part of the coordinate file to another force field can be readily performed. Actually many all-atom force fields share atom ordering, which often removes the need for such conversion. In addition to the atomistic CHARMM force field, CHARMM-GUI now also includes membrane builders for Martini [126] and PACE

[127] force fields. An example of a membrane protein system generated by CHARMM-GUI is shown in Fig. 1.

In this section the web-based membrane protein orientation tools and databases [128–131] should also be mentioned as they provide important information on how to position and align the proteins for the embedding methodologies that do not solve the protein alignment and positioning themselves. Contrary to the Protein Data Bank (PDB) [110], these databases provide complete membrane protein structures in which missing atoms are provided. Further, proteins are provided in their complete functional form where all symmetric subunits are reconstructed from the information in the original Protein Data Bank files, if needed. Notably, CHARMM-GUI can directly download oriented and complete protein structures from the OPM database [128,129].

5. Molecular dynamics simulation packages

The ease of building a membrane or the quality of the underlying force fields are superfluous if we cannot accurately perform a long enough simulation using the desired molecular ensemble. Molecular mechanics mostly relies on the ergodic assumption stating that in the limit of long simulation times, time averages match ensemble averages. Clearly, the longer we simulate, the higher are the chances that our simulation fulfills the ergodic assumption, stating that the whole configuration space is visited properly. It is not surprising that the simulation community tightly relies on a few highly advanced and complex MD packages and the talented developers behind them. With constantly increasing complexity of the computational resources paradigms (by *e.g.* GPUs, SIMD, openMP, MPI…) and the need to fully exploit their capabilities to the limit to tackle biologically relevant problems, the possibility that each researcher creates their own MD engine vanishes. Instead, most of the currently available MD packages are the result of decades of development by large communities. Their leaders are to thank for the golden age of membrane simulations we currently live in.

5.1. MD packages available for membrane simulations

Until recently most of the works involving membrane simulations have used either GROMACS [132] or NAMD [133] simulation packages. The main reason behind this is that they are the fastest available MD packages for such simulations [134]. However, things are changing as other MD packages are quickly catching up in performance, and therefore their use for membrane simulations is expected to grow steadily [134]. In particular, CHARMM [134,135], Amber [136,137], OpenMM [138] and LAMPPS [139] have growing user bases in the field.

Another reason that determines which MD package to use is the availability of force fields. Some MD packages, such as CHARMM and Amber, originally only supported their own force fields, *e.g.* CHARMM [135] and Amber [136]. The ones that do not provide their own force fields had to import third-party ones or provide a mechanism to use topology files from other MD packages. These strategies are also a fundamental reason behind the preferential usage of GROMACS and NAMD for membrane simulations as they supplied a large set of force fields to choose from. For these reasons other MD packages have followed this example to various degrees, *e.g. via* CHAMBER [140]. Currently, it is very difficult to know which force fields are fully compatible with each MD package. The instructions provided by the packages and third-party conversion tools are often confusing and very technical at best. This is becoming a real problem for users and for science itself as it leads to hard-to-track errors in the simulations. This could be overcome if the developers realized the criticality of this problem, and therefore agreed on some sort of a universally supported standard format for topology files.

Not all the MD packages provide the same functionality. For example, GROMACS offers a rather limited selection of free energy methods and advanced sampling algorithms when compared to Amber, NAMD or CHARMM. Importantly, the need to use one of these advanced methods also often dictates the choice of the MD package. Third-party software is aiming to tackle this very problem by providing generic implementations of these methods that can be easily used together with many MD packages. One well known example is the PLUMED library [141] that provides a wide variety of free energy and enhanced sampling methods compatible with a large amount of collective variable options. Although this strategy in not ideal from the performance point of view, it has many advantages. To begin with, it reduces the implementation time of an algorithm as this work only needs to be done once to provide the functionality to numerous MD packages. Additionally, the approach of PLUMED provides a good workaround to provide new features to some MD packages whose proprietary license (such as Amber, CHARMM, amd NAMD) restricts developers from modifying the code and redistributing their work.

Finally, another aspect of MD packages worth considering is the input and output formats, notably the format of the trajectory file. Available analysis tools can generally only deal with one or in the best case a few formats. Therefore, in case the aim is to perform a very particular or complex analysis on the simulation data (see Sections 6 and 7), the compatibility of the output formats with the analysis tool becomes an important factor affecting the choice of the MD package.

5.2. Are results dependent on the used MD package?

In the past, every MD package was able to handle a limited set of force fields and algorithms, which precluded comparison between them. With the increasing availability of the same force field and algorithms in different MD packages, the consistency of results between MD packages can finally be evaluated. As outlined above, MD packages are very complex pieces of software and the precise implementation of the numerous algorithms together with propagation of errors might affect the obtained results for a particular system. Moreover, not all the MD packages implement the same algorithms.

During the last few years increasing evidence has surfaced suggesting that a force field is not only the functional form and associated parameters of the interaction between atoms, but also the used algorithms together with their implementation details, which might vary between MD packages [115]. It is clear that only porting a force field without carefully considering these factors might lead to fundamentally different behavior. As an example, GROMACS version 4.6 [142] and earlier did not provide the cut-off schemes employed in CHARMM36 force field [66]. A careful attempt to overcome this discrepancy by optimizing simulation parameters was made [104], yet a satisfactory match was not obtained. What is alarming is that despite the availability of what seems to be the same algorithms (such as the CHARMM36-compatible cut-off schemes in GROMACS versions 5 and up), differences in behavior of lipid membranes still emerge due to implementation details that are extremely hard to track [115]. Finally, it should be noted that generally the time from the first implementation of a force field in one software [66] to it being properly tested in other ones might be substantially long [115], and the validity of the results published during this time span must be taken with a grain of salt.

6. Analysis of membranes

In this section we review the commonly employed tools for analyzing membrane properties. We leave for the next section some protein analysis tools which we consider to be useful for the analysis of membrane protein simulations. The tools are divided into sections based on their nature. First, most popular visualization tools are listed. This is followed by the tools and analysis interfaces provided with the MD simulation packages. Third, the available external and independent tools for trajectory post-processing are reviewed.

6.1. Visualization tools

A key feature of the MD method is the ability to visualize the motion of the molecules along the simulated trajectory. Average properties should always be extracted from a long enough simulation to provide meaningful information about the system. However, visual observation of the trajectory is a powerful tool that can often guide the analysis to the relevant properties of the system.

Several applications such as VMD [143], RasMol [144], [145], Avogadro [102] and PyMOL [146] provide the visualization capabilities for MD trajectories. All this software is freely available for academic purposes under a wide variety of licenses.

6.2. Tools bundled with the MD simulation software

Each of the common software packages used for MD simulations of biomolecular systems contains a number of analysis tools or at least a library allowing the user to generate such tools. There are also major differences in their ease-of-use as well as the extent of these bundled tools. Their biggest asset is that they natively read the files generated by the corresponding MD package.

The GROMACS package [142,132], for example, comes bundled with a large variety of tools covering both analyses of protein and membrane properties including energetics, structure and dynamics. Since GROMACS is provided under the LGPL license, the tools can be easily modified and redistributed, if the provided implementation does not provide exactly the required functionality. The built-in tools can also be supplemented by user-generated tools based on C (GROMACS v. 4.6 and below) or C++ (GROMACS v. 5 and above). A fully documented template for implementing user tools is provided. Tools generated with this template provide the same level of compatibility and efficiency as the built-in tools. These include native input/output support for GROMACS compatible files, access to large collection of efficient libraries for trajectory manipulation and data extraction, and even built-in parallelization capabilities. An important advantage of the GROMACS tools' philosophy is that they are highly optimized for memory usage, usually processing the trajectory data on-the-fly instead of storing it into memory. This is an important feature when one takes into consideration the ever increasing size of trajectories that is surpassing the amount of available memory in most desktop computers. Analysis tools in GROMACS are based on somewhat cumbersome atom selections defined prior to the analysis. On the other hand, this also allows the codes to be reused in quite different scenarios even beyond the original design. The most recent versions of GROMACS also support dynamic atom selections, which further improves the flexibility of the tools. The output files of default GROMACS analysis tools are compatible with the Grace plotting tool (usually operated via the graphical user interface xmgrace). Interestingly, there are several ways to use some GROMACS analysis capabilities from within Python, such as GromacsWrapper or the newly created official API. Worth mentioning here are also the tools developed for the calculation of pressure profiles [147–149]. These tools are based on modified GROMACS source code, and therefore serve as a good example of the adaptability provided by and open source code. When it comes to the calculation of pressure profiles, care must still be taken to avoid a large number of pitfalls [150,148].

The analysis tools of the NAMD simulation package [133] are provided in the VMD software [143]. In addition to its great visualization capabilities, VMD also provides analysis tools via its graphical user interface. This is supplemented by the Tk console, which can be used to execute the built-in tools as well as user-generated ones written in either Tcl or Python (the latter with limited support). Importantly, VMD provides powerful and dynamic atom selections. A large collection of user-generated tools is also readily available online, most notably on the VMD website. Tk console also allows interactive analyses. What is more, the capabilities of VMD extend beyond trajectories obtained from NAMD as VMD reads multiple file formats including those generated by all common MD packages considered in Section 5. One shortcoming of the philosophy of VMD-based analysis is that the whole analyzed trajectory is usually first loaded into memory, which limits its usability with large simulations on a regular desktop. However, the HiMach parallel analysis framework [151] overcomes this issue and allows the analysis of massive trajectories with VMD.

The CHARMM [135] simulation package also provides native analysis capabilities. To take advantage of these modules, the specific CHARMM scripting language must be used. Due to the restrictive license of CHARMM, modifying the tools to suit specific needs is not an easy task and it even requires an explicit permission from the authors.

The PTRAJ and CPPTRAJ programs [152] provide an interface to the common analysis tools for the Amber simulation package [153]. Additionally, they supply a framework for the user-generated tools. Whereas PTRAJ was written in C and launched already in 1990, the CPPTRAJ provides an updated version rewritten in C++, which aims to improve the performance of the original implementation.

The analysis functionality for LAMMPS [154] is provided by the pizza.py package (available at http://pizza.sandia.gov). Analysis can be performed either via an interactive interface or via scripts, and interfaces to popular visualization and plotting tools such as Matlab [155], GnuPlot [156], and VMD [143] are provided.

6.3. Individual analysis software

In addition to the tools supplied with or tightly linked to the simulation packages, a number of external software for the analysis of both proteins and membranes exist. Nowadays many of these tools aim for compatibility with the file formats associated with multiple MD packages. However, some software still relies on a single trajectory format. In such cases, the Open Babel software [103] provides an easy conversion between all common formats.

LOOS [157] is an object oriented library for the generation of analysis tools. LOOS is compatible with file types associated with multiple software including Amber, CHARMM, GROMACS and NAMD, among others. LOOS provides a large number of tools with dynamic atom selections for the analysis of both proteins and membrane systems. What is more, the library is intended to provide an easy way for the user to extend the provided functionality. LOOS is written in C++ for speed, while functionality can also be added through the Python interface called PyLOOS.

The g_lomepro tool [158] (from “local membrane properties”) analyzes the spatial dependence of some key membrane properties including lipid tail order parameters, membrane thickness, area per lipid and membrane curvature. The tool also handles systems containing membrane proteins making it suitable for the analysis of protein-induced changes in membrane structure. The tool is written in and built on the GROMACS framework, and therefore only file formats produced by GROMACS are supported. The tool is supplied with Perl scripts that aid the generation of videos showing temporal evolution of the studied membrane properties. An example of the 3D thickness map generated with g_lomepro is shown in Fig. 2.

Similarly, the GridMAT-MD tool [159] aims to study membrane properties on a 2-dimensional grid. It is a simple Perl tool that calculates membrane thickness and area per lipid maps. It supports GROMACS file formats.

The very recent Membrainy tool [160], written in Java to work out-of-the-box on numerous platforms, analyzes a range of membrane properties. It calculates both ensemble averages as well as resolves membrane properties on a grid. It analyzes lipid tail order parameters, membrane thickness and thickness maps, head group orientation, fraction of gel phase lipids, area per lipid and lipid mixing entropy, among some others. The tool is designed to provide the easiest possible access to most commonly analyzed membrane properties. Therefore, it might be complicated to apply Membrainy to tasks slightly different from what it was designed for. It is currently limited to reading only

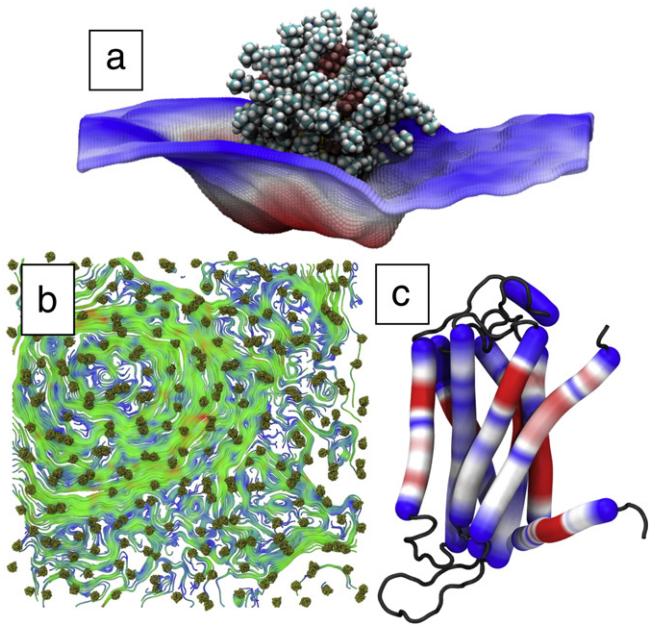

Fig. 2. Examples of advanced analysis tools. a) Output of the g_lomepro tool [158] shows the locally resolved thickness map upon gold nanoparticle adsorption to a membrane. The tool also resolves area per lipid, curvature and lipid tail order parameters in the same fashion. This figure is kindly provided by Fabio Lolicato. b) The Flows tool [171], built on top of the MDAnalysis framework [170], provides eye-pleasing visualization of the motion of lipids in membranes as well as other geometries. Here a vortex is observed in lipid motion in a system containing numerous membrane proteins (brown). This figure is courtesy of John E. Stone and Matthieu Chavent. c) The Bendix tool [181], now available as a VMD plugin, provides information on the local bending of alpha helices of proteins. It also supplies functionality to resolve the temporal evolution of helix bending. Here a static structure of the A_{2A} receptor is analyzed.

GROMACS file formats (as well as pdb files), and it relies on hard-coded support for lipid force fields. These currently include CHARMM36, Berger/GROMOS87 and Martini v2.0.

The MEMBPLUGIN [161] tool brings the ability to analyze local membrane properties to VMD. It is written in Tcl and provides a graphical user interface while command line execution is also supported. The tool calculates lipid tail order parameters, thickness maps, lipid tail interdigitation, area per lipid maps and tilt angles. The possibility to manually select the atoms used in the analyses provides more flexibility and due to the broad support for trajectory formats in VMD, MEMBPLUGIN can be used with all common force fields and file formats.

The APL@Voro tool [162] calculates membrane thickness and area per lipid locally, with the latter based on Voronoi tesselation (hence the name). The tool, written in C++, provides both a command line functionality as well as a graphical user interface with plotting capabilities. APL@Voro is compatible with Gromacs file formats.

JGROMACS [163] is a Java-based API enabling easy implementation of analysis tools with the capability to read GROMACS file formats. Additionally, some analysis functionality is included.

6.4. Analysis libraries for computational tools

Some analysis tools aim to take advantage of the great deal of algorithms provided by the existing generic high level programming languages. Such tools are built mainly on top of Matlab [155], Tcl, R [164] and Python. Matlab is a commercial software, whereas the other ones are available under different open source licenses and are, therefore, free of charge. Notably, Python has recently received lots of attention as numerous projects aiming to provide trajectory handling capabilities and common analysis routines, run on it. This is a natural choice since such tools can be built on top of the advanced scientific computing libraries NumPy [165] and SciPy [166]. What is more, the extensive 2D and 3D plotting capabilities provided by matplotlib [167] and Mayavi [168] further support the role of Python as the optimal platform for analysis tool generation.

In addition to an actual MD engine, the Molecular Modelling Toolkit (MMTK) [169] provides some analysis tools for proteins and routines for Amber/CHARMM trajectory processing. However, it is very limited, and therefore the more modern approaches (see below) should be followed.

Two very popular Python libraries for trajectory analysis have been recently introduced. Based on Python 2 and cython, the MDAnalysis toolkit [170] provides tools for analysis of both protein and membrane properties. MDAnalysis can read all common formats, including CHARMM, Amber, GROMACS, LAMMPS and NAMD topologies and trajectories. Efficient atom selection methods are also provided. Working on top of the MDAnalysis framework, the Flows tool [171] provides an interesting and visually compelling way to analyze lipid motion. An example of the output of this tool is shown in Fig. 2.

The very recent MDTraj [172] library also provides an interesting selection of trajectory handling capabilities and analysis routines. This software can be used in both Python v3 and Python v2. In order to improve its performance, several parts of the code are written in cython. MDTraj provides comprehensive file format reading capabilities similar to those of MDAnalysis as well as efficient selection routines. MDTraj currently supplies routines only for protein analysis.

The pmx Python library [173] (formerly known as pymacs) provides functionality to read and modify both structure and trajectory files in GROMACS formats. The library provides classes for efficient selection and indexing of residues and atoms yet analysis tools must be written by the user as they are not included by default. Importantly, pmx can also read and modify topology files as well as structure files of a number of Amber, OPLS-AA and CHARMM force fields, which can be efficiently exploited in setting up free energy calculations (see below).

MDToolbox [174] provides analysis tools that run on top of Matlab [155] or Octave [175]. By default, MDToolbox provides some functionality for analyzing proteins as well as free energy simulations. It includes functions that read Amber and CHARMM/NAMD trajectory and structure files. Structure files of GROMACS can also be read and the support for its binary trajectory formats is planned. Notably, the gro2mat tool [176] already provides the interface for reading the GROMACS trajectory data into Matlab.

The Bio3D package [177] runs on top of the R environment [164] and provides analysis tools to study protein conformations. Since it provides the trajectory reading capability for Amber file formats, new tools to extend functionality beyond protein conformations can easily be built using the powerful libraries provided by R.

7. Analysis of membrane proteins

Here we list some tools that can be employed for membrane-embedded proteins. First, most MD packages contain tools or functionality for protein analyses. Additionally, many analysis frameworks listed in the previous section, such as LOOS [157], MMTK [169], and MDAnalysis [170] provide tools for both membrane and protein analyses. Finally, some frameworks that provide trajectory reading and atom selection functionality contain built-in tools only for protein analyses. These include MDTraj [172], Bio3D [177], and MDToolbox [174] discussed in the previous section. In addition to these, we would like to mention a few independent tools here.

A web-based network analysis tool is provided by the MDN [178] portal. It allows the study of protein function from MD simulations. As an example, signal propagation in proteins can be examined via coupling in the generated network.

Wordom [179,180] is a tool for analyzing protein structure and dynamics from MD simulations. It contains tools for network analysis, principal component analysis, solvent accessible surface area calculation,

and secondary structure determination. The tool is written in C but also provides a Python interface.

The Bendix tool [181], now also a built-in feature of VMD, analyzes the bending of helices in static structures and dynamic trajectories. It provides both visualizations like that shown in Fig. 2 as well as temporal data on evolution of helix conformations. The tool works with both coarse-grained and atomistic models and thanks to the broad support of VMD, file formats of all common MD packages are supported.

The carma software [182] and the graphical user interface implementation grcarma [183] provide tools for the analysis of the structure and dynamics of proteins. The dcd file format is supported making carma/grcarma readily compatible with CHARMM and NAMD. Carma and grcarma are written in C and Perl/Tk, respectively.

8. Enhanced sampling and free energy techniques

In numerous cases the studied system is unable to sample the important parts of the phase space in the limited simulation time due to high energy barriers separating the relevant conformations of the system. What follows is that the post-processing analysis technique fails, and other approaches must be employed to accelerate the sampling. Such enhanced sampling methods include metadynamics, various replica exchange methods as well as simulated annealing, which all require specific runtime functionality. Luckily, the common MD packages natively provide at least some of this functionality.

In case the simulated system is unable to sample the desired conformations, one is often interested in the free energy barriers separating these conformations. Numerous free energy methods, including slow growth methods, thermodynamic integration (TI), and free energy perturbation aim to overcome these limitations and provide as a result the excess free energy profile as a function of the selected reaction coordinate. This reaction coordinate can be either alchemical or geometrical. In the latter case, the tools are often referred to as potential of mean force (PMF) methods (see Ref. [184]), and they include (adaptive) umbrella sampling as well as steered MD. Notably, these two are examples of an equilibrium and a non-equilibrium method, respectively. In addition, metadynamics can also provide the free energy profiles along with the acceleration in sampling. At least some of the listed free energy methods are provided with all common MD packages.

The development of the enhanced sampling and free energy algorithms is nowadays very fast and the MD packages usually lag behind in implementing these new features. PLUMED [185,141] provides a library that rapidly adapts the newest introduced techniques. It is written in C++ and designed to work together with most of the common MD simulation packages. Thanks to this universality, the tool has attracted a large user community. An active discussion forum is a major advantage that PLUMED has over free energy implementations specific to certain MD engines. PLUMED provides tools for PMF calculations including metadynamics, umbrella sampling and steered MD. Notably, different kinds of bias potentials can be applied to an extensive collection of collective variables. METAGUI [186] is a VMD plugin designed to provide a graphical user interface for analyzing metadynamics simulations performed with PLUMED.

Additionally, some tools aim for the easy setup of free energy calculations. The FESetup [187] Python tool provides an easy way to setup TI calculations for both Amber and GROMACS simulation programs. It is compatible with Amber force fields.

The pmx tool mentioned above as an analysis tool can also be employed to setup free energy calculations for GROMACS using various force fields [173,188]. The tool is designed to generate topologies corresponding to amino acid mutations which can then be used with TI and other methods to calculate the associated free energy changes.

The highly mobile membrane-mimetic (HMMM) model [189] replaces lipid tails by small organic molecules, which results in one or two orders of magnitude larger lipid diffusion. This approach significantly reduces the time scale required to study adsorption of peripheral proteins, and could, therefore, be considered as an enhanced sampling method, even though the Hamiltonian is not altered. Recently, the capability of constructing systems for HMMM simulations was included in CHARMM-GUI [190].

9. Conclusions and future directions

This review provides the state of the art of the tools associated with molecular dynamics simulations of membranes and membrane proteins. We began by introducing the most commonly employed lipid and protein force fields. Both of these have seen significant improvement in their accuracy during this last decade. In addition, the number of supported lipid moieties has been constantly growing. We have also pointed out that some of the lipid force fields were especially designed to be compatible with protein force fields, hence allowing accurate membrane simulations containing proteins. Such systems are fundamental biological environments, and therefore, these recent improvements in computational models commence to allow the combination of simulation and experiment to tackle key questions in cellular membrane biology.

However, we feel that the current way of force field development is far from optimal. First of all, the numerous simulation programs provide different algorithms and their implementations. This results in the dependence of the outcome on the employed MD package. Therefore, introduction of methodologies that would eliminate such dependence, such as the cut-off free Lennard-Jones potential, would be highly appreciated. This naturally comes with the cost of reparameterizing the force fields. However, this effort might still pay off in the long term.

We also think that the constant demand for more accurate force fields would be best satisfied if the generation of lipid parameters could be somehow automated. Such an automated platform would be of great importance as more and more lipid parameters as well as parameters for other types of molecules are required to account for the vast diversity of real biological membranes.

We then reviewed the tools available for building lipid membranes. This procedure has shifted from error-prone manual approaches towards newly emerged user-friendly tools that are capable to construct custom complex membranes in a highly automated fashion. Yet many of these membrane builder tools are only compatible with a very small spectrum of force fields, lipid environments, and MD packages. Furthermore, we listed approaches for embedding membrane proteins into lipid membranes. Numerous new methods have been recently introduced to take care of this procedure in a way that would cause minimal perturbation to the host membrane, which helps to minimize the equilibration phase after protein insertion. Some of the approaches aim to provide a very general embedding method independent of force fields, file formats, and external software, yet they often fall short in terms of ease-of-use. On the other hand, the straightforward approaches are often limited to work with few force fields and simulation platforms, and might also require installation of additional external software. What is more, most approaches were designed to assemble lipids in planar geometries. However, these tools will need to be extended as simulations of vesicles, bicelles or even lipid droplets become more common.

We feel that the highly automated tools for setting up MD simulations are important as they can be readily adapted by newcomers in the field. On the other hand, membrane simulation veterans still benefit from the adaptability of some seemingly more complicated approaches. Therefore, both approaches are required to satisfy the ever growing membrane and membrane protein simulation community.

Finally, we have discussed the emergence of new analysis tools that allow researchers to study membranes and membrane proteins more efficiently and employ more creative approaches. These tools are often openly available, which allows researchers to dedicate more time to the actual science rather than to tool development.

Analysis programs are usually slow even to the point at which some analyses on their output trajectories are more time consuming than the well parallelized simulations themselves. Currently available analysis tools are usually not well parallelized yet rather use just one core or a few at best. This is clearly an area where we will see rapid improvement in the future cherished by the consolidation of development of universal analysis environments which aim to be independent of the MD packages, and therefore of file formats. Hardware development will also have a critical role in the analysis of MD simulations. Modern desktop computers are equipped with multi-core processors, which allows the use of several cores simultaneously, thus drastically decreasing the time required for analyses. Another point worth mentioning is the amount of memory, as several analyses require the whole trajectory to be stored in memory at once. This is a real issue when considering the size of the trajectories produced already nowadays, let alone in the future as developments in computing power will make the situation much worse.

Finally, we would like to point out that only few of the listed tools are versatile enough to be applied to other than planar membranes with small fluctuations. As simulated membranes grow in size, undulations will increase drastically, which calls for smarter approaches for various tasks related to analysis. These include *e.g.*, separation of the membrane into leaflets, determination of the local membrane normal, and calculation of displacements along curved surfaces, to name a few. Again, analysis of vesicles, bicelles, or lipid droplets would also benefit greatly from the development of more flexible and smart tools that would not rely on oversimplified assumptions.

These reviewed developments, even though still very scattered and highly specific, clearly point towards a brighter future. The ultimate goal clearly is to have fully automated, user-friendly tools and error-free protocols that support all kinds of imaginable membranes interacting with all kinds of molecules. Furthermore, automated molecule parameterization, and intelligent, fast as well as universal analysis tools are also desired.

Though in our opinion we are still far from this ultimate goal, we could possibly speed up our journey there by following a few guidelines. First, we need to encourage discussion between the developers of the different MD packages so that file format compatibility stops being an issue like it is today. Perhaps it is time to consider adapting a common standard for the file formats, even though this would require major work from the developers. In the best case this could result in the compatibility of every MD package with numerous membrane builder and protein embedding tools as well as analysis software. This would substantially enlarge the toolkits of researchers in the field. What is more, this would allow easier cross-checking of the results provided by different MD packages.

Second, we need to promote initiatives and options that limit unnecessary repetition of certain tasks. Nowadays, as seen from the extensive list of tools above, several implementations of the same analysis algorithm are frequently encountered often without clear improvement between them. With universally adapted file formats, each algorithm should require only one implementation unless new features or substantial improvements in performance are added in new implementations in the future. This second point, in our opinion, will require all scientific software to be open source and covered with flexible licenses that allow anybody to contribute to the code and freely redistribute the resulting version of the tool.

Summarizing, during the last years the membrane simulation field has profited extensively from development of software. Indeed, performing a membrane simulation is nowadays a rather simple task if compared to the situation a decade ago. This attracts researchers to engage in computational membrane biophysics and membrane protein biophysics, and therefore further contribute to the advance of the field *via* the development of tools, models and force fields. However, we must still educate these people of the possible pitfalls involved in the modern seemingly straightforward software. Overall, we are living in the golden age of membrane and membrane-protein simulations and we are confident that many great advances are still to appear and consolidate the status of computational biophysics as an important counterpart to traditional laboratory experiments.

Transparency Document

The Transparency document associated with this article can be found, in online version.

Acknowledgments

We would like to thank Dr. Giray Enkavi and Dr. Maria Kalimeri for the critical discussions about the content. We acknowledge financial support from the Academy of Finland through its Centre of Excellence Programme.

References

[1] H. Scott, Monte Carlo studies of lipid/water interfaces, BBA-Biomembr. 814 (2) (1985) 327–332.
[2] J. Wendoloski, S. Kimatian, C. Schutt, F. Salemme, Molecular dynamics simulation of a phospholipid micelle, Science 243 (4891) (1989) 636–638.
[3] P.S. Charifson, R.G. Hiskey, L.G. Pedersen, Construction and molecular modeling of phospholipid surfaces, J. Comput. Chem. 11 (10) (1990) 1181–1186.
[4] M.L. Berkowitz, K. Raghavan, Computer simulation of a water/membrane interface, Langmuir 7 (6) (1991) 1042–1044.
[5] T.B. Woolf, B. Roux, Molecular dynamics simulation of the gramicidin channel in a phospholipid bilayer, Proc. Natl. Acad. Sci. U. S. A. 91 (24) (1994) 11631–11635.
[6] O. Edholm, O. Berger, F. Jähnig, Structure and fluctuations of bacteriorhodopsin in the purple membrane: a molecular dynamics study, J. Mol. Biol. 250 (1) (1995) 94–111.
[7] T. Róg, M. Pasenkiewicz-Gierula, I. Vattulainen, M. Karttunen, Ordering effects of cholesterol and its analogues, BBA-Biomembr. 1788 (1) (2009) 97–121.
[8] P.S. Niemelä, M.T. Hyvönen, I. Vattulainen, Atom-scale molecular interactions in lipid raft mixtures, BBA-Biomembr. 1788 (1) (2009) 122–135.
[9] P.C. Biggin, P.J. Bond, Molecular dynamics simulations of membrane proteins, Molecular Modeling of Proteins, Springer 2015, pp. 91–108.
[10] T. Baştuğ, S. Kuyucak, Molecular dynamics simulations of membrane proteins, Biophys. Rev. 4 (3) (2012) 271–282.
[11] K. Pluhackova, T.A. Wassenaar, R.A. Böckmann, Molecular dynamics simulations of membrane proteins, Membrane Biogenesis, Springer 2013, pp. 85–101.
[12] C. Kandt, W.L. Ash, D.P. Tieleman, Setting up and running molecular dynamics simulations of membrane proteins, Methods 41 (4) (2007) 475–488.
[13] H. Martinez-Seara, T. Róg, Molecular dynamics simulations of lipid bilayers: simple recipe of how to do it, Methods Mol. Biol. 924 (2013) 407–429.
[14] A. Kukol, Lipid membranes for membrane proteins, Molecular Modeling of Proteins, Springer 2015, pp. 73–90.
[15] K. Pluhackova, R.A. Böckmann, Biomembranes in atomistic and coarse-grained simulations, J. Phys. Condens. Matter 27 (32) (2015) 323103.
[16] P.E. Lopes, O. Guvench, A.D. MacKerell Jr., Current status of protein force fields for molecular dynamics simulations, Molecular Modeling of Proteins, Springer 2015, pp. 47–71.
[17] A. Botan, F. Favela-Rosales, P.F.J. Fuchs, M. Javanainen, Mf. Kanduč, W. Kulig, A. Lamberg, C. Loison, A. Lyubartsev, M.S. Miettinen, L. Monticelli, J. Määttä, O.H.S. Ollila, M. Retegan, T. Róg, H. Santuz, J. Tynkkynen, Toward atomistic resolution structure of phosphatidylcholine headgroup and glycerol backbone at different ambient conditions, J. Phys. Chem. B 119 (49) (2015) 15075–15088.
[18] E.A. Cino, W.-Y. Choy, M. Karttunen, Comparison of secondary structure formation using 10 different force fields in microsecond molecular dynamics simulations, J. Chem. Theory Comput. 8 (8) (2012) 2725–2740.
[19] K. Lindorff-Larsen, P. Maragakis, S. Piana, M.P. Eastwood, R.O. Dror, D.E. Shaw, Systematic validation of protein force fields against experimental data, PLoS One 7 (2) (2012), e32131.
[20] S.J. Marrink, H.J. Risselada, S. Yefimov, D.P. Tieleman, A.H. De Vries, The MARTINI force field: coarse grained model for biomolecular simulations, J. Phys. Chem. B 111 (27) (2007) 7812–7824.
[21] C.A. López, Z. Sovova, F.J. van Eerden, A.H. de Vries, S.J. Marrink, Martini force field parameters for glycolipids, J. Chem. Theory Comput. 9 (3) (2013) 1694–1708.
[22] L. Monticelli, S.K. Kandasamy, X. Periole, R.G. Larson, D.P. Tieleman, S.-J. Marrink, The MARTINI coarse-grained force field: extension to proteins, J. Chem. Theory Comput. 4 (5) (2008) 819–834.
[23] D.H. de Jong, G. Singh, W.D. Bennett, C. Arnarez, T.A. Wassenaar, L.V. Schafer, X. Periole, D.P. Tieleman, S.J. Marrink, Improved parameters for the martini coarse-grained protein force field, J. Chem. Theory Comput. 9 (1) (2012) 687–697.
[24] C.A. López, A.J. Rzepiela, A.H. De Vries, L. Dijkhuizen, P.H. Hunenberger, S.J. Marrink, Martini coarse-grained force field: extension to carbohydrates, J. Chem. Theory Comput. 5 (12) (2009) 3195–3210.
[25] J.J. Uusitalo, H.I. Ingólfsson, P. Akhshi, D.P. Tieleman, S.J. Marrink, Martini coarse-grained force field: extension to DNA, J. Chem. Theory Comput. 11 (8) (2015) 3932–3945.

[26] S.O. Yesylevskyy, L.V. Schäfer, D. Sengupta, S.J. Marrink, Polarizable water model for the coarse-grained MARTINI force field, PLoS Comput. Biol. 6 (6) (2010), e1000810.
[27] C. Arnarez, J.J. Uusitalo, M.F. Masman, H.I. Ingólfsson, D.H. de Jong, M.N. Melo, X. Periole, A.H. De Vries, S.J. Marrink, Dry Martini, a coarse-grained force field for lipid membrane simulations with implicit solvent, J. Chem. Theory Comput. 11 (1) (2014) 260–275.
[28] T. Bereau, M. Deserno, Generic coarse-grained model for protein folding and aggregation, J. Chem. Phys. 130 (23) (2009) 235106.
[29] Z.-J. Wang, M. Deserno, A systematically coarse-grained solvent-free model for quantitative phospholipid bilayer simulations, J. Phys. Chem. B 114 (34) (2010) 11207–11220.
[30] T. Bereau, Z.-J. Wang, M. Deserno, More than the sum of its parts: coarse-grained peptide-lipid interactions from a simple cross-parametrization, J. Chem. Phys. 140 (11) (2014) 115101.
[31] M. Orsi, J.W. Essex, The ELBA force field for coarse-grain modeling of lipid membranes, PLoS One 6 (12) (2011), e28637.
[32] N. Schmid, A.P. Eichenberger, A. Choutko, S. Riniker, M. Winger, A.E. Mark, W.F. van Gunsteren, Definition and testing of the GROMOS force-field versions 54 A7 and 54B7, Eur. Biophys. J. 40 (7) (2011) 843–856.
[33] S.-W. Chiu, S.A. Pandit, H. Scott, E. Jakobsson, An improved united atom force field for simulation of mixed lipid bilayers, J. Phys. Chem. B 113 (9) (2009) 2748–2763.
[34] A. Kukol, Lipid models for united-atom molecular dynamics simulations of proteins, J. Chem. Theory Comput. 5 (3) (2009) 615–626.
[35] D. Poger, W.F. Van Gunsteren, A.E. Mark, A new force field for simulating phosphatidylcholine bilayers, J. Comput. Chem. 31 (6) (2010) 1117–1125.
[36] D.M. van Aalten, R. Bywater, J.B. Findlay, M. Hendlich, R.W. Hooft, G. Vriend, PRODRG, a program for generating molecular topologies and unique molecular descriptors from coordinates of small molecules, J. Comput. Aided Mol. Des. 10 (3) (1996) 255–262.
[37] A.W. Schüttelkopf, D.M. Van Aalten, PRODRG: a tool for high-throughput crystallography of protein–ligand complexes, Acta Crystallogr. Sect. D Biol. Crystallogr. 60 (8) (2004) 1355–1363.
[38] J.A. Lemkul, W.J. Allen, D.R. Bevan, Practical considerations for building GROMOS-compatible small-molecule topologies, J. Chem. Inf. Model. 50 (12) (2010) 2221–2235.
[39] A.K. Malde, L. Zuo, M. Breeze, M. Stroet, D. Poger, P.C. Nair, C. Oostenbrink, A.E. Mark, An automated force field topology builder (ATB) and repository: version 1.0, J. Chem. Theory Comput. 7 (12) (2011) 4026–4037.
[40] K.B. Koziara, M. Stroet, A.K. Malde, A.E. Mark, Testing and validation of the Automated Topology Builder (ATB) version 2.0: prediction of hydration free enthalpies, J. Comput. Aided Mol. Des. 28 (3) (2014) 221–233.
[41] S. Canzar, M. El-Kebir, R. Pool, K. Elbassioni, A.K. Malde, A.E. Mark, D.P. Geerke, L. Stougie, G.W. Klau, Charge group partitioning in biomolecular simulation, J. Comp. Biol. 20 (3) (2013) 188–198.
[42] O. Berger, O. Edholm, F. Jähnig, Molecular dynamics simulations of a fluid bilayer of dipalmitoylphosphatidylcholine at full hydration, constant pressure, and constant temperature, Biophys. J. 72 (5) (1997) 2002.
[43] R. Tjörnhammar, O. Edholm, Reparameterized united atom model for molecular dynamics simulations of gel and fluid phosphatidylcholine bilayers, J. Chem. Theory Comput. 10 (12) (2014) 5706–5715.
[44] N. Bhatnagar, G. Kamath, J.J. Potoff, Biomolecular simulations with the transferable potentials for phase equilibria: extension to phospholipids, J. Phys. Chem. B 117 (34) (2013) 9910–9921.
[45] K. Lindorff-Larsen, S. Piana, K. Palmo, P. Maragakis, J.L. Klepeis, R.O. Dror, D.E. Shaw, Improved side-chain torsion potentials for the Amber ff99SB protein force field, Proteins Struct. Funct. Bioinf. 78 (8) (2010) 1950–1958.
[46] J.A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K.E. Hauser, C. Simmerling, ff14SB: improving the accuracy of protein side chain and backbone parameters from ff99SB, J. Chem. Theory Comput. 11 (8) (2015) 3696–3713.
[47] D.S. Cerutti, W.C. Swope, J.E. Rice, D.A. Case, ff14ipq: a self-consistent force field for condensed-phase simulations of proteins, J. Chem. Theory Comput. 10 (10) (2014) 4515–4534.
[48] Y. Duan, C. Wu, S. Chowdhury, M.C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, et al., A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations, J. Comput. Chem. 24 (16) (2003) 1999–2012.
[49] C.J. Dickson, L. Rosso, R.M. Betz, R.C. Walker, I.R. Gould, GAFFlipid: a General Amber Force Field for the accurate molecular dynamics simulation of phospholipid, Soft Matter 8 (37) (2012) 9617–9627.
[50] Å.A. Skjevik, B.D. Madej, R.C. Walker, K. Teigen, LIPID11: a modular framework for lipid simulations using amber, J. Phys. Chem. B 116 (36) (2012) 11124–11136.
[51] C.J. Dickson, B.D. Madej, Å.A. Skjevik, R.M. Betz, K. Teigen, I.R. Gould, R.C. Walker, Lipid14: the amber lipid force field, J. Chem. Theory Comput. 10 (2) (2014) 865–879.
[52] B.D. Madej, I.R. Gould, R.C. Walker, A parameterization of cholesterol for mixed lipid bilayer simulation within the Amber Lipid14 force field, J. Phys. Chem. B 119 (38) (2015) 12424–12435.
[53] J.P. Jämbeck, A.P. Lyubartsev, Derivation and systematic validation of a refined all-atom force field for phosphatidylcholine lipids, J. Phys. Chem. B 116 (10) (2012) 3164–3179.
[54] J.P. Jämbeck, A.P. Lyubartsev, An extension and further validation of an all-atomistic force field for biological membranes, J. Chem. Theory Comput. 8 (8) (2012) 2938–2948.
[55] J.P. Jämbeck, A.P. Lyubartsev, Another piece of the membrane puzzle: extending slipids further, J. Chem. Theory Comput. 9 (1) (2012) 774–784.
[56] K.N. Kirschner, A.B. Yongye, S.M. Tschampel, J. González-Outeiriño, C.R. Daniels, B.L. Foley, R.J. Woods, GLYCAM06: a generalizable biomolecular force field. Carbohydrates, J. Comput. Chem. 29 (4) (2008) 622–655.
[57] J. Wang, R.M. Wolf, J.W. Caldwell, P.A. Kollman, D.A. Case, Development and testing of a general amber force field, J. Comput. Chem. 25 (9) (2004) 1157–1174.
[58] J. Wang, W. Wang, P.A. Kollman, D.A. Case, Automatic atom type and bond type perception in molecular mechanical calculations, J. Mol. Graphics Modell. 25 (2) (2006) 247–260.
[59] L. Huang, B. Roux, Automated force field parameterization for nonpolarizable and polarizable atomic models based on ab initio target data, J. Chem. Theory Comput. 9 (8) (2013) 3543–3556.
[60] R.M. Betz, R.C. Walker, Paramfit: automated optimization of force field parameters for molecular dynamics simulations, J. Comput. Chem. 36 (2) (2015) 79–87.
[61] J. Wang, W. Wang, P.A. Kollman, D.A. Case, Antechamber: an accessory software package for molecular mechanical calculations, J. Am. Chem. Soc. 222 (2001) U403.
[62] D. van der Spoel, P.J. van Maaren, C. Caleman, GROMACS molecule & liquid database, Bioinformatics 28 (5) (2012) 752–753.
[63] A.D. MacKerell, D. Bashford, M. Bellott, R. Dunbrack, J. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S.A. Ha, et al., All-atom empirical potential for molecular modeling and dynamics studies of proteins, J. Phys. Chem. B 102 (18) (1998) 3586–3616.
[64] A.D. MacKerell, M. Feig, C.L. Brooks, Extending the treatment of backbone energetics in protein force fields: limitations of gas-phase quantum mechanics in reproducing protein conformational distributions in molecular dynamics simulations, J. Comput. Chem. 25 (11) (2004) 1400–1415.
[65] R.B. Best, X. Zhu, J. Shim, P.E. Lopes, J. Mittal, M. Feig, A.D. MacKerell Jr., Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone ϕ, ψ and side-chain $\chi1$ and $\chi2$ dihedral angles, J. Chem. Theory Comput. 8 (9) (2012) 3257–3273.
[66] J.B. Klauda, R.M. Venable, J.A. Freites, J.W. O'Connor, D.J. Tobias, C. Mondragon-Ramirez, I. Vorobyov, A.D. MacKerell Jr., R.W. Pastor, Update of the CHARMM all-atom additive force field for lipids: validation on six lipid types, J. Phys. Chem. B 114 (23) (2010) 7830–7843.
[67] R.M. Venable, A.J. Sodt, B. Rogaski, H. Rui, E. Hatcher, A.D. MacKerell, R.W. Pastor, J.B. Klauda, Charmm all-atom additive force field for sphingomyelin: elucidation of hydrogen bonding and of positive curvature, Biophys. J. 107 (1) (2014) 134–145.
[68] O. Guvench, S.S. Mallajosyula, E.P. Raman, E. Hatcher, K. Vanommeslaeghe, T.J. Foster, F.W. Jamison, A.D. MacKerell Jr., CHARMM additive all-atom force field for carbohydrate derivatives and its utility in polysaccharide and carbohydrate–protein modeling, J. Chem. Theory Comput. 7 (10) (2011) 3162–3180.
[69] K. Hart, N. Foloppe, C.M. Baker, E.J. Denning, L. Nilsson, A.D. MacKerell Jr., Optimization of the CHARMM additive force field for DNA: improved treatment of the BI/BII conformational equilibrium, J. Chem. Theory Comput. 8 (1) (2011) 348–362.
[70] J.D. Yesselman, D.J. Price, J.L. Knight, C.L. Brooks, MATCH: an atom-typing toolset for molecular mechanics force fields, J. Comput. Chem. 33 (2) (2012) 189–202.
[71] K. Vanommeslaeghe, E. Hatcher, C. Acharya, S. Kundu, S. Zhong, J. Shim, E. Darian, O. Guvench, P. Lopes, I. Vorobyov, et al., CHARMM general force field: a force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields, J. Comput. Chem. 31 (4) (2010) 671–690.
[72] K. Vanommeslaeghe, E.P. Raman, A.D. MacKerell Jr., Automation of the CHARMM General Force Field (CGenFF) II: assignment of bonded parameters and partial atomic charges, J. Chem. Inf. Model. 52 (12) (2012) 3155–3168.
[73] K. Vanommeslaeghe, A.D. MacKerell Jr., Automation of the CHARMM General Force Field (CGenFF) I: bond perception and atom typing, J. Chem. Inf. Model. 52 (12) (2012) 3144–3154.
[74] W.L. Jorgensen, J. Tirado-Rives, The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin, J. Am. Chem. Soc. 110 (6) (1988) 1657–1666.
[75] M.J. Robertson, J. Tirado-Rives, W.L. Jorgensen, Improved peptide and protein torsional energetics with the OPLS-AA force field, J. Chem. Theory Comput. 11 (7) (2015) 3499–3509.
[76] A. Maciejewski, M. Pasenkiewicz-Gierula, O. Cramariuc, I. Vattulainen, T. Rog, Refined OPLS all-atom force field for saturated phosphatidylcholine bilayers at full hydration, J. Phys. Chem. B 118 (17) (2014) 4571–4581.
[77] W. Kulig, M. Pasenkiewicz-Gierula, T. Róg, Cis and trans unsaturated phosphatidylcholine bilayers: a molecular dynamics simulation study, Chem. Phys. Lipids 195 (2016) 12–20.
[78] D. Kony, W. Damm, S. Stoll, W.F. Van Gunsteren, An improved OPLS–AA force field for carbohydrates, J. Comput. Chem. 23 (15) (2002) 1416–1429.
[79] A. Cordom, G. Caltabiano, L. Pardo, Membrane protein simulations using AMBER force field and Berger lipid parameters, J. Chem. Theory Comput. 8 (3) (2012) 948–958.
[80] D.P. Tieleman, J.L. MacCallum, W.L. Ash, C. Kandt, Z. Xu, L. Monticelli, Membrane protein simulations with a united-atom lipid and all-atom protein model: lipid–protein interactions, side chain transfer free energies and model proteins, J. Phys. Condens. Matter 18 (28) (2006) S1221.
[81] W. Han, C.-K. Wan, F. Jiang, Y.-D. Wu, PACE force field for protein simulations. 1. full parameterization of version 1 and verification, J. Chem. Theory Comput. 6 (11) (2010) 3373–3389.
[82] W. Han, C.-K. Wan, Y.-D. Wu, PACE force field for protein simulations. 2. folding simulations of peptides, J. Chem. Theory Comput. 6 (11) (2010) 3390–3402.
[83] C.-K. Wan, W. Han, Y.-D. Wu, Parameterization of pace force field for membrane environment and simulation of helical peptides and helix–helix association, J. Chem. Theory Comput. 8 (1) (2011) 300–313.

[84] Q. Shi, S. Izvekov, G.A. Voth, Mixed atomistic and coarse-grained molecular dynamics: simulation of a membrane-bound ion channel, J. Phys. Chem. B 110 (31) (2006) 15045–15048.
[85] T.A. Wassenaar, H.I. Ingólfsson, M. Priess, S.J. Marrink, L.V. Schäfer, Mixing MARTINI: electrostatic coupling in hybrid atomistic-coarse-grained biomolecular simulations, J. Phys. Chem. B 117 (13) (2013) 3516–3530.
[86] J. Chowdhary, E. Harder, P.E. Lopes, L. Huang, A.D. MacKerell Jr., B. Roux, A polarizable force field of dipalmitoylphosphatidylcholine based on the classical drude model for molecular dynamics simulations of lipids, J. Phys. Chem. B 117 (31) (2013) 9142–9160.
[87] P.E. Lopes, J. Huang, J. Shim, Y. Luo, H. Li, B. Roux, A.D. MacKerell Jr., Polarizable force field for peptides and proteins based on the classical Drude oscillator, J. Chem. Theory Comput. 9 (12) (2013) 5430–5449.
[88] C.M. Baker, V.M. Anisimov, A.D. MacKerell Jr., Development of CHARMM polarizable force field for nucleic acid bases based on the classical Drude oscillator model, J. Phys. Chem. B 115 (3) (2010) 580–596.
[89] C.M. Baker, Polarizable force fields for molecular dynamics simulations of biomolecules, Wiley Interdiscip. Rev. Comput. Mol. Sci. 5 (2) (2015) 241–254.
[90] J. Huang, P.E. Lopes, B. Roux, A.D. MacKerell Jr., Recent advances in polarizable force fields for macromolecules: microsecond simulations of proteins using the classical Drude oscillator model, J. Phys. Chem. Lett. 5 (18) (2014) 3144–3150.
[91] K. Vanommeslaeghe, A. MacKerell, CHARMM additive and polarizable force fields for biophysics and computer-aided drug design, BBA-Biomembr. 1850 (5) (2015) 861–871.
[92] P. Brocos, P. Mendoza-Espinosa, R. Castillo, J. Mas-Oliva, Á. Pineiro, Multiscale molecular dynamics simulations of micelles: coarse-grain for self-assembly and atomic resolution for finer details, Soft Matter 8 (34) (2012) 9005–9014.
[93] T.A. Wassenaar, K. Pluhackova, R.A. Bockmann, S.J. Marrink, D.P. Tieleman, Going backward: a flexible geometric approach to reverse transformation from coarse grained to atomistic models, J. Chem. Theory Comput. 10 (2) (2014) 676–690.
[94] P.J. Stansfeld, M.S. Sansom, From coarse grained to atomistic: a serial multiscale approach to membrane protein simulations, J. Chem. Theory Comput. 7 (4) (2011) 1157–1166.
[95] P. Larsson, P.M. Kasson, Lipid converter, a framework for lipid manipulations in molecular dynamics simulations, J. Membr. Biol. 247 (11) (2014) 1137–1140.
[96] D. Marsh, Handbook of lipid bilayers, second ed. CRC Press, 2013, ISBN 978-1-4200-8833-5.
[97] H. Koldsø, M.S. Sansom, Organization and dynamics of receptor proteins in a plasma membrane, J. Am. Chem. Soc. 137 (46) (2015) 14694–14704.
[98] H.I. Ingólfsson, M.N. Melo, F.J. van Eerden, C. Arnarez, C.A. Lopez, T.A. Wassenaar, X. Periole, A.H. De Vries, D.P. Tieleman, S.J. Marrink, Lipid organization of the plasma membrane, J. Am. Chem. Soc. 136 (41) (2014) 14554–14559.
[99] R. Ramon Guixà-González, M. Javanainen, M. Gómez-Soler, B.n. Cordobilla, J.C. Domingo, F. Sanz, M. Pastor, F. Ciruela, H. Martinez-Seara, J. Selent, Membrane omega-3 fatty acids modulate the oligomerisation kinetics of adenosine A2A and dopamine D2 receptors, Sci. Rep. 6 (2016) 19839.
[100] A.Y. Antipina, A.A. Gurtovenko, Molecular mechanism of calcium-induced adsorption of DNA on zwitterionic phospholipid membranes, J. Phys. Chem. B 119 (22) (2015) 6638–6645.
[101] C. Bovigny, G. Tamo, T. Lemmin, N. Mano, M. Dal Peraro, LipidBuilder: a framework to build realistic models for biological membranes, J. Chem. Inf. Model. 55 (12) (2015) 2491–2499.
[102] M.D. Hanwell, D.E. Curtis, D.C. Lonie, T. Vandermeersch, E. Zurek, G.R. Hutchison, Avogadro: an advanced semantic chemical editor, visualization, and analysis platform, J. Cheminform. 4 (1) (2012) 17.
[103] N.M. OLBoyle, M. Banck, C.A. James, C. Morley, T. Vandermeersch, G.R. Hutchison, Open babel: an open chemical toolbox, J. Cheminform. 3 (2011) 33.
[104] T.J. Piggot, Á. Piñeiro, S. Khalid, Molecular dynamics simulations of phosphatidylcholine membranes: a comparative force field study, J. Chem. Theory Comput. 8 (11) (2012) 4593–4609.
[105] J. Domański, P.J. Stansfeld, M.S. Sansom, O. Beckstein, Lipidbook: a public repository for force-field parameters used in membrane simulations, J. Membr. Biol. 236 (3) (2010) 255–258.
[106] E.L. Wu, X. Cheng, S. Jo, H. Rui, K.C. Song, E.M. Dávila-Contreras, Y. Qi, J. Lee, V. Monje-Galvan, R.M. Venable, J.B. Klauda, W. Im, CHARMM-GUI membrane builder toward realistic biological membrane simulations, J. Comput. Chem. 35 (27) (2014) 1997–2004.
[107] M. Javanainen, Universal method for embedding proteins into complex lipid bilayers for molecular dynamics simulations, J. Chem. Theory Comput. 10 (6) (2014) 2577–2582.
[108] H. Martinez-Seara, T. Róg, M. Karttunen, I. Vattulainen, R. Reigada, Why is the sn-2 chain of monounsaturated glycerophospholipids usually unsaturated whereas the sn-1 chain is saturated? Studies of 1-stearoyl-2-oleoyl-sn-glycero-3-phosphatidylcholine (SOPC) and 1-oleoyl-2-stearoyl-sn-glycero-3-phosphatidylcholine (OSPC) membranes with and without cholesterol, J. Phys. Chem. B 113 (24) (2009) 8347–8356.
[109] C. Hong, D.P. Tieleman, Y. Wang, Microsecond molecular dynamics simulations of lipid mixing, Langmuir 30 (40) (2014) 11993–12001.
[110] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, P. Bourne, The Protein Data Bank, Nucleic Acids Res. 28 (1) (2000) 235–242.
[111] S. Jo, J.B. Lim, J.B. Klauda, W. Im, CHARMM-GUI membrane builder for mixed bilayers and its application to yeast membranes, Biophys. J. 97 (1) (2009) 50–58.
[112] M.M. Ghahremanpour, S.S. Arab, S.B. Aghazadeh, J. Zhang, D. van der Spoel, MemBuilder: a web-based graphical interface to build heterogeneously mixed membrane bilayers for the GROMACS biomolecular simulation program, Bioinformatics 30 (3) (2013) 439–441.
[113] L. Martnez, R. Andrade, E.G. Birgin, J.M. Martnez, PACKMOL: a package for building initial configurations for molecular dynamics simulations, J. Comput. Chem. 30 (13) (2009) 2157–2164.
[114] T.A. Wassenaar, H.I. Ingólfsson, R.A. Bockmann, D.P. Tieleman, S.J. Marrink, Computational lipidomics with insane: a versatile tool for generating custom membranes for molecular simulations, J. Chem. Theory Comput. 11 (5) (2015) 2144–2155.
[115] J. Lee, X. Cheng, J.M. Swails, M.S. Yeom, P.K. Eastman, J.A. Lemkul, S. Wei, J. Buckner, J.C. Jeong, Y. Qi, CHARMM-GUI input generator for NAMD, GROMACS, AMBER, OpenMM, and CHARMM/OpenMM simulations using the CHARMM36 additive force field, J. Chem. Theory Comput. 12 (1) (2016) 405–413.
[116] L. Shen, D. Bassolino, T. Stouch, Transmembrane helix structure, dynamics, and interactions: multi-nanosecond molecular dynamics simulations, Biophys. J. 73 (1) (1997) 3.
[117] J.D. Faraldo-Gómez, G.R. Smith, M.S. Sansom, Setting up and optimization of membrane protein simulations, Eur. Biophys. J. 31 (3) (2002) 217–227.
[118] T.H. Schmidt, C. Kandt, LAMBADA and InflateGRO2: efficient membrane alignment and insertion of membrane proteins for molecular dynamics simulations, J. Chem. Inf. Model. 52 (10) (2012) 2657–2669.
[119] M.G. Wolf, M. Hoefling, C. Aponte-Santamara, H. Grubmüller, G. Groenhof, g_membed: efficient insertion of a membrane protein into an equilibrated lipid bilayer with minimal perturbation, J. Comput. Chem. 31 (11) (2010) 2169–2174.
[120] R. Staritzbichler, C. Anselmi, L.R. Forrest, J.D. Faraldo-Gómez, GRIFFIN: a versatile methodology for optimization of protein–lipid interfaces for membrane protein simulations, J. Chem. Theory Comput. 7 (4) (2011) 1167–1176.
[121] T.A. Wassenaar, K. Pluhackova, A. Moussatova, D. Sengupta, S.J. Marrink, D.P. Tieleman, R.A. Böckmann, High-throughput simulations of dimer and trimer assembly of membrane proteins. the DAFT approach, J. Chem. Theory Comput. 11 (5) (2015) 2278–2291.
[122] E. Jefferys, Z.A. Sands, J. Shi, M.S. Sansom, P.W. Fowler, Alchembed: a computational method for incorporating multiple proteins into complex lipid geometries, J. Chem. Theory Comput. 11 (6) (2015) 2743–2754.
[123] P.J. Stansfeld, J.E. Goose, M. Caffrey, E.P. Carpenter, J.L. Parker, S. Newstead, M.S. Sansom, MemProtMD: automated insertion of membrane protein structures into explicit lipid membranes, Structure 23 (7) (2015) 1350–1361.
[124] S. Jo, T. Kim, V.G. Iyer, W. Im, CHARMM-GUI: a web-based graphical user interface for CHARMM, J. Comput. Chem. 29 (11) (2008) 1859–1865.
[125] S. Jo, T. Kim, W. Im, Automated builder and database of protein/membrane complexes for molecular dynamics simulations, PLoS One 2 (9) (2007), e880.
[126] Y. Qi, H.I. Ingólfsson, X. Cheng, J. Lee, S.J. Marrink, W. Im, CHARMM-GUI Martini maker for coarse-grained simulations with the Martini force field, J. Chem. Theory Comput. 11 (9) (2015) 4486–4494.
[127] Y. Qi, X. Cheng, W. Han, S. Jo, K. Schulten, W. Im, CHARMM-GUI PACE CG builder for solution, micelle, and bilayer coarse-grained simulations, J. Chem. Inf. Model. 54 (3) (2014) 1003–1009.
[128] M.A. Lomize, A.L. Lomize, I.D. Pogozheva, H.I. Mosberg, Opm: orientations of proteins in membranes database, Bioinformatics 22 (5) (2006) 623–625.
[129] M.A. Lomize, I.D. Pogozheva, H. Joo, H.I. Mosberg, A.L. Lomize, OPM database and PPM web server: resources for positioning of proteins in membranes, Nucleic Acids Res. 40 (D1) (2012) D370–D376.
[130] T. Kimmett, N. Smith, S. Witham, M. Petukh, S. Sarkar, E. Alexov, ProBLM web server: protein and membrane placement and orientation package, Comput. Math. Methods Med. (2014).
[131] T. Nugent, D.T. Jones, Membrane protein orientation and refinement using a knowledge-based statistical potential, BMC Bioinf. 14 (1) (2013) 276.
[132] M.J. Abraham, T. Murtola, R. Schulz, S. Páll, J.C. Smith, B. Hess, E. Lindahl, GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers, SoftwareX 1–2 (2015) 19–25.
[133] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kale, K. Schulten, Scalable molecular dynamics with NAMD, J. Comput. Chem. 26 (16) (2005) 1781–1802.
[134] A.-P. Hynninen, M.F. Crowley, New faster CHARMM molecular dynamics engine, J. Comput. Chem. 35 (5) (2013) 406–413.
[135] B.R. Brooks, C.L. Brooks, A.D. Mackerell, L. Nilsson, R.J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A.R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R.W. Pastor, C.B. Post, J.Z. Pu, M. Schaefer, B. Tidor, R.M. Venable, H.L. Woodcock, X. Wu, W. Yang, D.M. York, M. Karplus, CHARMM: the biomolecular simulation program, J. Comput. Chem. 30 (10) (2009) 1545–1614.
[136] D. Case, J. Berryman, R. Betz, D. Cerutti, T. Cheatham III, T. Darden, R. Duke, T. Giese, H. Gohlke, A. Goetz, et al., AMBER, University of California, San Francisco, 2015.
[137] R. Salomon-Ferrer, D.A. Case, R.C. Walker, An overview of the Amber biomolecular simulation package, Wiley Interdiscip. Rev. Comput. Mol. Sci. 3 (2) (2012) 198–210.
[138] P. Eastman, M.S. Friedrichs, J.D. Chodera, R.J. Radmer, C.M. Bruns, J.P. Ku, K.A. Beauchamp, T.J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M.R. Shirts, V.S. Pande, OpenMM 4: a reusable, extensible, hardware independent library for high performance molecular simulation, J. Chem. Theory Comput. 9 (1) (2013) 461–469.
[139] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[140] M.F. Crowley, M.J. Williamson, R.C. Walker, CHAMBER: comprehensive support for CHARMM force fields within the AMBER software, Int. J. Quantum Chem. 109 (15) (2009) 3767–3772.
[141] G.A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, G. Bussi, PLUMED 2: new feathers for an old bird, Comput. Phys. Commun. 185 (2) (2014) 604–613.

[142] S. Pronk, S. Pall, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M.R. Shirts, J.C. Smith, P.M. Kasson, D. van der Spoel, B. Hess, E. Lindahl, GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit, Bioinformatics 29 (7) (2013) 845–854.
[143] W. Humphrey, A. Dalke, K. Schulten, VMD: visual molecular dynamics, J. Mol. Graph. 14 (1) (1996) 33–38.
[144] R.A. Sayle, E.J. Milner-White, RASMOL: biomolecular graphics for all, Trends Biochem. Sci. 20 (9) (1995) 374.
[145] D.S. Goodsell, Representing Structural Information with RasMol, John Wiley & Sons, Inc., 2002 (Ch. UNIT 5.4).
[146] L.L.C. Schrödinger, The PyMOL Molecular Graphics System, version 1.3r1, August 2010.
[147] O.S. Ollila, H.J. Risselada, M. Louhivuori, E. Lindahl, I. Vattulainen, S.J. Marrink, 3D pressure field in lipid membranes and membrane–protein complexes, Phys. Rev. Lett. 102 (7) (2009) 078101.
[148] J.M. Vanegas, A. Torres-Sánchez, M. Arroyo, Importance of force decomposition for local stress calculations in biomembrane molecular simulations, J. Chem. Theory Comput. 10 (2) (2014) 691–702.
[149] A. Torres-Sánchez, J.M. Vanegas, M. Arroyo, Examining the mechanical equilibrium of microscopic stresses in molecular simulations, Phys. Rev. Lett. 114 (25) (2015) 258102.
[150] J. Sonne, F.Y. Hansen, G.H. Peters, Methodological problems in pressure profile calculations for lipid bilayers, J. Chem. Phys. 122 (12) (2005) 124903.
[151] T. Tu, C. Rendleman, D.W. Borhani, R.O. Dror, J. Gullingsrud, M.Ø. Jensen, J.L. Klepeis, P. Maragakis, P. Miller, K. Stafford, et al., A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories, High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008, International Conference for, IEEE 2008, pp. 1–12.
[152] D.R. Roe, T.E. Cheatham III, PTRAJ and CPPTRAJ: software for processing and analysis of molecular dynamics trajectory data, J. Chem. Theory Comput. 9 (7) (2013) 3084–3095.
[153] D.A. Case, T.E. Cheatham, T. Darden, H. Gohlke, R. Luo, K.M. Merz, A. Onufriev, C. Simmerling, B. Wang, R.J. Woods, The Amber biomolecular simulation programs, J. Comput. Chem. 26 (16) (2005) 1668–1688.
[154] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[155] The Mathworks, Inc., Natick, Massachusetts, MATLAB version 8.5.0.197613 (R2015a), 2015.
[156] T. Williams, C. Kelley, R. Lang, D. Kotz, J. Campbell, G. Elber, A. Woo, et al., Gnuplot 5: an interactive plotting program, June 2015 (URL http://www.gnuplot.info).
[157] T.D. Romo, N. Leioatts, A. Grossfield, Lightweight object oriented structure analysis: Tools for building tools to analyze molecular dynamics simulations, J. Comput. Chem. 35 (32) (2014) 2305–2318.
[158] V. Gapsys, B.L. de Groot, R. Briones, Computational analysis of local membrane properties, J. Comput. Aided Mol. Des. 27 (10) (2013) 845–858.
[159] W.J. Allen, J.A. Lemkul, D.R. Bevan, GridMAT-MD: a grid-based membrane analysis tool for use with molecular dynamics, J. Comput. Chem. 30 (12) (2009) 1952–1958.
[160] M. Carr, C.E. MacPhee, Membrainy: a 'smart', unified membrane analysis tool, Source Code Biol. Med. 10 (1) (2015) 3.
[161] R. Guixà-González, I. Rodriguez-Espigares, J.M. Ramrez-Anguita, P. Carrió-Gaspar, H. Martinez-Seara, T. Giorgino, J. Selent, MEMBPLUGIN: studying membrane complexity in VMD, Bioinformatics 30 (10) (2014) 1478–1480.
[162] G. Lukat, J. Krüger, B. Sommer, APL@ Voro: a voronoi-based membrane analysis tool for GROMACS trajectories, J. Chem. Inf. Model. 53 (11) (2013) 2908–2925.
[163] M. Münz, P.C. Biggin, JGromacs: a Java package for analyzing protein simulations, J. Chem. Inf. Model. 52 (1) (2012) 255–259.
[164] R Core Team, R: a language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, 2015 (https://www.R-project.org).
[165] S. Van Der Walt, S.C. Colbert, G. Varoquaux, The NumPy array: a structure for efficient numerical computation, IEEE Comput. Sci. Eng. 13 (2) (2011) 22–30.
[166] E. Jones, T. Oliphant, P. Peterson, SciPy: open source scientific tools for Python, 2001 (Online; accessed 2016-03-04) http://www.scipy.org/.
[167] J.D. Hunter, Matplotlib: a 2D graphics environment, IEEE Comput. Sci. Eng. 9 (3) (2007) 90–95.
[168] P. Ramachandran, G. Varoquaux, Mayavi: 3D visualization of scientific data, IEEE Comput. Sci. Eng. 13 (2) (2011) 40–51.
[169] K. Hinsen, The molecular modeling toolkit: a new approach to molecular simulations, J. Comput. Chem. 21 (2) (2000) 79–85.
[170] N. Michaud-Agrawal, E.J. Denning, T.B. Woolf, O. Beckstein, MDAnalysis: a toolkit for the analysis of molecular dynamics simulations, J. Comput. Chem. 32 (10) (2011) 2319–2327.
[171] M. Chavent, T. Reddy, J. Goose, A.C.E. Dahl, J.E. Stone, B. Jobard, M.S. Sansom, Methodologies for the analysis of instantaneous lipid diffusion in md simulations of large membrane systems, Faraday Discuss. 169 (2014) 455–475.
[172] R.T. McGibbon, K.A. Beauchamp, M.P. Harrigan, C. Klein, J.M. Swails, C.X. Hernández, C.R. Schwantes, L.-P. Wang, T.J. Lane, V.S. Pande, MDTraj: a modern open library for the analysis of molecular dynamics trajectories, Biophys. J. 109 (8) (2015) 1528–1532.
[173] V. Gapsys, S. Michielssens, D. Seeliger, B.L. de Groot, pmx: automated protein structure and topology generation for alchemical perturbations, J. Comput. Chem. 36 (5) (2015) 348–354.
[174] Y. Matsunaga, mdtoolbox documentation, August 2015 URL http://mdtoolbox.readthedocs.org/mdtoolbox.
[175] J.W. Eaton, D. Bateman, S. Hauberg, R. Wehbring, GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations, 2015 URL http://www.gnu.org/software/octave/doc/interpreter.
[176] H. Dien, C.M. Deane, B. Knapp, Gro2mat: a package to efficiently read Gromacs output in Matlab, J. Comput. Chem. 35 (20) (2014) 1528–1531.
[177] B.J. Grant, A.P. Rodrigues, K.M. ElSawy, J.A. McCammon, L.S. Caves, Bio3d: an R package for the comparative analysis of protein structures, Bioinformatics 22 (21) (2006) 2695–2696.
[178] A.A. Ribeiro, V. Ortiz, MDN: a web portal for network analysis of molecular dynamics simulations, Biophys. J. 109 (6) (2015) 1110–1116.
[179] M. Seeber, M. Cecchini, F. Rao, G. Settanni, A. Caflisch, Wordom: a program for efficient analysis of molecular dynamics simulations, Bioinformatics 23 (19) (2007) 2625–2627.
[180] M. Seeber, A. Felline, F. Raimondi, S. Muff, R. Friedman, F. Rao, A. Caflisch, F. Fanelli, Wordom: a user-friendly program for the analysis of molecular structures, trajectories, and free energy surfaces, J. Comput. Chem. 32 (6) (2011) 1183–1194.
[181] A.C.E. Dahl, M. Chavent, M.S. Sansom, Bendix: intuitive helix geometry analysis and abstraction, Bioinformatics 28 (16) (2012) 2193–2194.
[182] N.M. Glykos, Software news and updates carma: a molecular dynamics analysis program, J. Comput. Chem. 27 (14) (2006) 1765–1768.
[183] P.I. Koukos, N.M. Glykos, Grcarma: a fully automated task-oriented interface for the analysis of molecular dynamics trajectories, J. Comput. Chem. 34 (26) (2013) 2310–2312.
[184] D. Trzesniak, A.-P.E. Kunz, W.F. van Gunsteren, a comparison of methods to compute the potential of mean force, Chem. Phys. Chem. 8 (1) (2007) 162–169.
[185] M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R.A. Broglia, et al., PLUMED: a portable plugin for free-energy calculations with molecular dynamics, Comput. Phys. Commun. 180 (10) (2009) 1961–1972.
[186] X. Biarnés, F. Pietrucci, F. Marinelli, A. Laio, METAGUI. a VMD interface for analyzing metadynamics and molecular dynamics simulations, Comput. Phys. Commun. 183 (1) (2012) 203–211.
[187] H.H. Loeffler, J. Michel, C.J. Woods, FESetup: automating setup for alchemical free energy simulations, J. Chem. Inf. Model. 55 (12) (2015) 2485–2490.
[188] D. Seeliger, B.L. De Groot, Protein thermostability calculations using alchemical free energy simulations, Biophys. J. 98 (10) (2010) 2309–2316.
[189] Y.Z. Ohkubo, T.V. Pogorelov, M.J. Arcario, G.A. Christensen, E. Tajkhorshid, Accelerating membrane insertion of peripheral proteins with a novel membrane mimetic model, Biophys. J. 102 (9) (2012) 2130–2139.
[190] Y. Qi, X. Cheng, J. Lee, J.V. Vermaas, T.V. Pogorelov, E. Tajkhorshid, S. Park, J.B. Klauda, W. Im, CHARMM-GUI HMMM builder for membrane simulations with the highly mobile membrane-mimetic model, Biophys. J. 109 (10) (2015) 2012–2022.