OrganL: Dynamic Triangulation of Biomembranes using Curved Elements

Christoph Allolio[1*], Balázs Fábián[2], and Mark Dostalík[1]

[1]Charles University, Faculty of Mathematics and Physics, Mathematical Institute, Sokolovská 83, 186 75 Prague 8, Czech Republic
[2]Institute of Organic Chemistry and Biochemistry of the Czech Academy of Sciences, Flemingovo nám. 542/2, Czech Republic
[*]Correspondence: allolio@karlin.mff.cuni.cz

ABSTRACT We describe a method for simulating biomembranes of arbitrary shape. In contrast to other dynamically triangulated surface (DTS) algorithms, our method provides a rich, *quasi* tangent-continuous, yet local description of the surface. We use curved Nagata triangles, which we generalize to cubic order to achieve the requisite flexibility. The resulting interpolation can be constructed locally without iterations, at the cost of having only approximate tangent continuity away from the vertices. This allows us to provide a parallelized and fine-tuned Monte Carlo implementation. As a first example of the potential benefits of the enhanced description, our method supports inhomogeneous lipid distributions as well as lipid mixing. It also supports restraints and constraints of various types and is constructed to be as easily extensible as possible. We validate the approach by testing its numerical accuracy, followed by reproducing the known Helfrich solutions for shapes with rotational symmetry. Finally, we present some example applications, including curvature-driven demixing and stylized effects of proteins. Input files for these examples, as well as the implementation itself, are freely available for researchers under the name OrganL.

Our method provides a straightforward way to simulate any biomembrane geometry. It overcomes some of the limitations of previous dynamically triangulated surface (DTS) Monte Carlo schemes by providing a surface that contains an interpolant which allows to assign meaningful functions of curvature to almost every point of the discretization, yet keeps much of the simplicity of the common DTS schemes by not requiring any nonlocal information or iterations for its construction. Our tool is easily extensible and facilitates the simulation of complex lipid and protein compositions on membrane surfaces at any scale.

INTRODUCTION

A realistic multiscale model of (sub)cellular structures, such as organelles remains a distant goal. Molecular dynamics (MD) simulations provide detailed information on molecular structure and surface binding but are limited by size and time scales. Coarse-grained (CG) MD simulations provide limited relief, as they still require to simulate particles of a roughly molecular size and often entail a substantial loss of accuracy and transferability.(1) The obvious starting point for the mesoscopic description of lipid membrane-based systems is provided by the (Evans-Canham)-Helfrich functional(2–4), which has found wide application in membrane biophysics(5–9) since its introduction in the seventies. Recently, efficient techniques to extract its local parameters from atomistic MD simulations have been developed.(10–14) Introducing molecular specificity into the continuum model in this way has the potential to allow the systematic construction of models for slow processes at the cellular scale and to solve them using modest computational means. An illustration was recently provided for membrane fusion.(11) The heterogeneous nature of biomembranes requires local and dynamic membrane properties. However, even in the absence of such concerns, the Helfrich functional is difficult to extremalize analytically as well as numerically. This is particularly true in the absence of symmetries, as surfaces are hard to parameterize in a general setting. Dynamically triangulated surface (DTS) simulations, as pioneered(15) and developed(16, 17) by Kroll and Gompper among others(18–20) are, therefore, often employed to explore static as well as time-dependent behavior of complex membrane structures.(21–24) The DTS method, as it is established, is based on an operator discretization over the vertices.(25) Such discretizations are widely used in established packages(26, 27), such as Brakke's surface evolver(28), but the underlying construction does not allow for any substructure on the faces, relegating all information to the vertices. This leads, for example, to the practice of confining membrane-deforming proteins to the mesh-vertices(27, 29) and makes it difficult to compute accurate interaction energies of membrane structures at curved interfaces. The insertion of complex mixing and coupling terms or flow fields thus requires a very fine mesh, even though actual changes in shape are often of large scale when compared to the protein size. It is, of course, possible to describe curvature in a continuous manner, using splines(30) and there are other known finite element approaches.(31) The Nagata interpolation(32) is *quasi* tangent continuous and, as we shall see, provides a sufficient

degree of accuracy in the calculation of the curvature integrals required for the Helfrich theory. However, it also has substantial drawbacks. In particular, the interpolation requires the provision of surface normals. Normal estimation on unstructured meshes remains a challenging problem and the standard Nagata interpolant fails to give reasonable results for a wide range of "unexpected" normals.(33) As the resolution of the problems of the Nagata interpolant requires modification of both the interpolant and commonly used Monte Carlo steps, the Methods section necessarily contains a significant amount of new methodology; in other words, it contains results.

METHODS

Discretization Scheme

The starting point of the Nagata interpolation scheme(32) is the search for a quadratic edge interpolant $\boldsymbol{\varphi}_2(t)$ between two vertices $\boldsymbol{x}_A, \boldsymbol{x}_B$ connected by a distance vector $\boldsymbol{d} = \boldsymbol{x}_B - \boldsymbol{x}_A$. We denote the (vector) coefficients of the interpolant with $\boldsymbol{c}_1$. Defining

$$\boldsymbol{\varphi}_2(t) = \boldsymbol{x}_A + (\boldsymbol{d} - \boldsymbol{c}_1)\, t + \boldsymbol{c}_1 t^2, \tag{1}$$

ensures that the interpolant goes to $\boldsymbol{x}_A$ at $t = 0$ and $\boldsymbol{x}_B$ at $t = 1$. Furthermore, if the coefficient vector $\boldsymbol{c}_1$ is zero, $\boldsymbol{\varphi}_2(t)$ becomes a straight line connecting the vertices. The key difference to other interpolation schemes is that instead of solving for the coefficients by some boundary conditions (such as equal derivative values) defined by the neighboring patches, the Nagata scheme uses normal vectors $\boldsymbol{n}_1, \boldsymbol{n}_2$ to determine the coefficients without any direct reference to neighboring patches. Reproducing surface normal vectors requires the knowledge of at least one more interpolated edge to construct the tangent basis. However, for simplicity, Nagata decided to merely consider tangent vectors of the single edge interpolant

$$\boldsymbol{t}(t) = \boldsymbol{\varphi}_2'(t) = \boldsymbol{d} - \boldsymbol{c}_1 + 2\boldsymbol{c}_1 t. \tag{2}$$

A consistent normal vector requires the tangents $\boldsymbol{t}_A = \boldsymbol{t}(0)$ and $\boldsymbol{t}_B = \boldsymbol{t}(1)$ at the endpoints to be orthogonal to the normal. Hence, the conditions

$$\boldsymbol{n}_A \cdot \boldsymbol{t}_A = \boldsymbol{n}_A \cdot (\boldsymbol{d} - \boldsymbol{c}_1) = 0, \tag{3a}$$
$$\boldsymbol{n}_B \cdot \boldsymbol{t}_B = \boldsymbol{n}_B \cdot (\boldsymbol{d} + \boldsymbol{c}_1) = 0, \tag{3b}$$

need to be fulfilled. However, $\boldsymbol{c}_1$ has three elements. Hence the resulting linear equation system $\mathbb{M}$ for $\boldsymbol{c}_1$ is underdetermined. As $\boldsymbol{c}_1$ controls the "curvature" of the interpolant, it makes sense to insert the minimization of $||\boldsymbol{c}_1||$ as the third condition to complete the system. Note that for $||\boldsymbol{c}_1|| > 0$, this norm has no straightforward relation to curvature as it is commonly defined on curves, as an analytical arc-length representation is not possible. The pseudoinverse, also known as the Moore-Penrose inverse of a matrix $\mathbb{M}$, is a generalization of the matrix inverse to nonsquare matrices. It guarantees fulfillment of $\mathbb{M}$, while minimizing $||\boldsymbol{c}_1||$ (least squares). Following reference (32), the pseudoinverse can be computed analytically via

$$\mathbb{M}^{+} = \lim_{\alpha \to 0^+} \left(\mathbb{M}^{\top}\mathbb{M} + \alpha \mathbb{I}\right)^{-1} \mathbb{M}^{\top}, \tag{4}$$

leading to simple expressions for the determination of $\boldsymbol{c}_1$. Fig. 1 shows the construction of an edge interpolant in a non-pathological case.

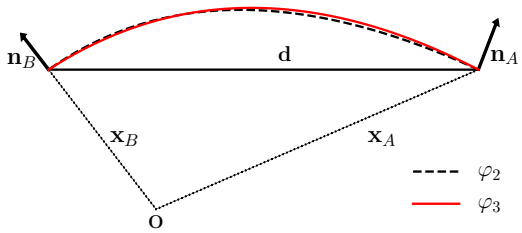

Figure 1: Quadratic (dotted line) and cubic (red line) Nagata edge interpolants, for the case of well-behaved normal vectors.

This process fails to give the expected results for a wide variety of pathological normal orientations. Singular points and their remedies have been discussed by Nagata.(32) For our purposes, it is sufficient that the interpolation will succeed for smooth surfaces, as is required by the ordinary Helfrich theory. It was discovered recently that the results for

$$(\boldsymbol{n}_A \cdot \boldsymbol{d})\,(\boldsymbol{n}_B \cdot \boldsymbol{d}) \geq 0, \tag{5}$$

are problematic.(33) Some of the issues are illustrated in Fig. 2. Namely, if both of the normal vectors are oriented in the same way with respect to $\boldsymbol{d}$, the quadratic interpolant will not lie in the angle between normal vectors viewed along $\boldsymbol{d}$. An example of this arrangement is given in the left panel of Fig. 2, where the two normals are approximately parallel and close to orthogonal to $\boldsymbol{d}$. Such a configuration of normal vectors may arise readily, e.g. from a normal estimation with a small perturbation of a flat mesh during the formation of a dimple. The resulting interpolant describes a large "sideways" bow, risking overlaps with other parts of the mesh. Similar problems occur in the event of twisting the normals as shown on the right of Fig. 2. If the interpolant were to lie within the space spanned by the normals and $\boldsymbol{d}$, it would have to contain an inflection point. For such cases, Nagata and subsequent authors advise either subdivide the mesh to cover the inflection point or to insert a flat line. In the case of a membrane simulation, such an approach is difficult to implement, as there are no underlying data to refine and a realistic expression of curvature is required. Furthermore, our experiments with updating the mesh with only quadratic patches were not successful. In order to ameliorate the situation, we increase the interpolation to cubic order:

$$\boldsymbol{\varphi}_3(t) = \boldsymbol{x}_A + (\boldsymbol{d} - \boldsymbol{c}_1 - \boldsymbol{c}_2)\, t + \boldsymbol{c}_1 t^2 + \boldsymbol{c}_2 t^3. \tag{6}$$

The derivative is given by

$$\boldsymbol{t} = \boldsymbol{\varphi}_3'(t) = \boldsymbol{d} - \boldsymbol{c}_1 - \boldsymbol{c}_2 + 2\boldsymbol{c}_1 t + 3\boldsymbol{c}_2 t^2. \tag{7}$$

As before, we impose the requirement of the tangent vectors to be perpendicular to the normals at the endpoints. The introduction of $\boldsymbol{c}_2$ means that the equation system now has four degrees of freedom left to determine. Note that, e.g. the arc-length variation of a curve between two points, for a given set of tangent vectors will result in a straight line with an infinitesimal kink; it is therefore expected that an arbitrary increase of polynomial order will lead to an equivalent result. In order to achieve a good interpolant, it is necessary to require additional conditions. For the pseudoinverse operation to work, these conditions have to come in the form of linear equations. Therefore, we impose an additional estimate of the binormal vector. This vector is orthogonal to the normal vector and is estimated to be normal to the plane spanned by $\boldsymbol{n}$ and $\boldsymbol{d}$, the construction is intended to keep $\boldsymbol{\varphi}_3'$ nearly parallel to $\boldsymbol{d}$. This leads to the following set of conditions:

$$\boldsymbol{n}_A \cdot \boldsymbol{t}_A = 0, \tag{8a}$$
$$\boldsymbol{n}_B \cdot \boldsymbol{t}_B = 0, \tag{8b}$$
$$(\boldsymbol{d} \times \boldsymbol{n}_A) \cdot \boldsymbol{t}_A = 0, \tag{8c}$$
$$(\boldsymbol{d} \times \boldsymbol{n}_B) \cdot \boldsymbol{t}_B = 0. \tag{8d}$$

As remarked previously, using the pseudoinverse for an underdetermined system corresponds to solving a least squares problem in the coefficient vector. This problem has an analytical solution, which we provide in the Supplementary Material, together with the detailed algorithm which produces the edge interpolant with a few vector multiplications. We find them to be well-behaved, resulting in an interpolant very close to the original Nagata formulation for the "nonpathological" cases as well as a reasonable inflection point for the failure mode described in Eq. (5). The additional condition introduced minimizes the chance of edge overlap.

Interpolation of a Patch

In order to create a surface interpolant, the edge interpolants have to be assembled into a surface. The patch interpolant $\boldsymbol{x}_2(\eta, \zeta)$ for the original quadratic Nagata edges is also of second order:

$$\boldsymbol{x}_2(\eta,\zeta) = \widetilde{\boldsymbol{c}_{00}} + \widetilde{\boldsymbol{c}_{10}}\eta + \widetilde{\boldsymbol{c}_{01}}\zeta + \widetilde{\boldsymbol{c}_{11}}\eta\zeta + \widetilde{\boldsymbol{c}_{20}}\eta^2 + \widetilde{\boldsymbol{c}_{02}}\zeta^2. \tag{9}$$

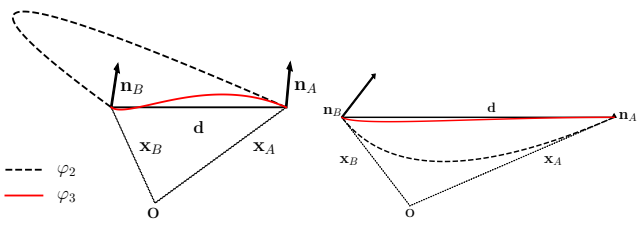

Figure 2: Left panel: Quadratic (dotted line) and cubic (red line) Nagata edge interpolants, for the case of normal vectors with equal orientation $\boldsymbol{n}_i \cdot \boldsymbol{d} > 0$. Right panel: Arrangement for twisted normals with the additional condition of $\boldsymbol{n}_2 \cdot \boldsymbol{d} \approx 0$.

The six coefficients of this polynomial are determined by the requirement of the edges of the surface interpolant being equal to the edge interpolants. More precisely, the functions $\boldsymbol{x}_2(\eta, 0)$, $\boldsymbol{x}_2(1, \zeta)$, $\boldsymbol{x}_2(\eta, \eta)$ have to be exactly equal to the Nagata edge interpolants. For a visualization of the structure, see Fig. 3. The edge polynomials restrict the domain of the interpolated patch to $0 \le \eta \le 1$ and $0 \le \zeta \le \eta$.

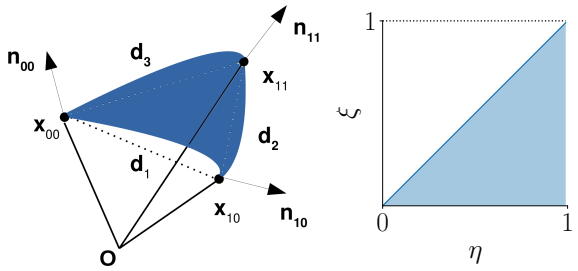

Figure 3: Left panel: Schematic of a patch interpolant as constructed from the edge interpolants. Right panel: Domain of the patch interpolant.

The third-order patch interpolant requires additional conditions to be uniquely determined. It must reprodcue the quadratic case when $\boldsymbol{c}_2 = 0$, resolving any ambiguity. A detailed derivation is given in the Supplementary Material. In our code, cubic edges are only used where the failure condition in Eq. (5) is met as the alternatives of permitting bad interpolants or inserting a straight line lead to heavy artifacts. Excluding these configurations leads to the necessity of creating complicated, collective motions of the mesh (clustered moves) to avoid trapping. The stand-alone accuracy of the resulting interpolant is evaluated in the Results section.

Evaluation of the Helfrich Energy

For a closed surface, the Helfrich free energy F, supplemented by surface tension σ and pressure p terms can be stated as

$$F = \int_{\text{Surf.}} \mathrm{dA} \left\{ \frac{\kappa}{2}(\tilde{H} - J_S)^2 + \bar{\kappa} K_G + \sigma \right\} + \int_{\text{Vol.}} \mathrm{dV} p. \tag{10}$$

In this equation, the geometrical quantities include the total curvature $\tilde{H}$ (the sum of principal curvatures, in contrast to the standard definition of the mean curvature $H = \frac{1}{2}\tilde{H}$), the Gaussian curvature K_G, the area element dA and the volume element dV. The material parameters are the bending rigidity κ, the Gaussian bending modulus $\bar{\kappa}$ and the spontaneous or intrinsic curvature J_S. The pressure p and surface tension σ can be interpreted as Lagrange multipliers on the total surface and volume, respectively. In our scheme, A and V are controlled instead by quadratic penalties and K_G can typically be ignored due to the Gauss-Bonnet theorem. We note, that V can be obtained from the divergence theorem using the spatial coordinate $\boldsymbol{r}$ as

$$V = \frac{1}{3}\int_{\text{Vol.}} \nabla \cdot \boldsymbol{r}\, \mathrm{dV} = \frac{1}{3}\int_{\text{Surf.}} \boldsymbol{x} \cdot \hat{\boldsymbol{n}}\, \mathrm{dA}, \tag{11}$$

where $\hat{\boldsymbol{n}}$ is the surface unit normal vector as computed from the patch interpolant $\boldsymbol{x}$. If the surface is closed, it is sufficient to work with surface integrals to evaluate all of F. The surface integral is summed over all patches and in each patch is discretized on the domain of the interpolant using a seventh-order Gaussian quadrature scheme(34). Note the facility of constructing parallel planes accounting explicitly for separate monolayers and the availability of a reasonable direct way to evaluate Gaussian curvature on a face. If the surface is open, an additional term

$$F_\Gamma = \int_{\partial\Gamma} \left[\overline{\kappa} k_G + \tau\right] \mathrm{d}\mathbf{l} \tag{12}$$

needs to be added, where the integration is performed along the boundary $\partial\Gamma$ of the surface, and a geodesic curvature k_G as well as line tension τ of the membrane need to be integrated along the line element $\mathrm{d}\mathbf{l}$. On the level of the code, different energy functionals are available via subclassing and all of the above described terms can be computed. Some further possibilities are described in the Supplementary Material, most notably an implementation of area difference elasticity (ADE)(35, 36) via control of the integrated mean curvature. Local curvatures can be expressed in a simplified manner by the coefficients. More details about the implementation are given in the Supplementary Material.

Parallelized MC Implementation

Moves

We use a standard Metropolis Monte-Carlo (MC) algorithm to evolve the interpolated surface and sample the Boltzmann distribution of the underlying energy functional. The advantage of this setup is that fluctuations, such as membrane undulations are automatically included in the simulation results. The sampling of a finite temperature distribution also implies our code is not an optimization code. In contrast to standard DTS, we also have to consider the changes of the normals in the sampling of equilibrium geometries. We provide vertex moves, normal moves and so-called deep vertex moves. In addition to this, we have also implemented a basic bond-flipping "Alexander" move and a new type of move associated with lipid composition on a face. Due to the local nature of these moves, the code is easily parallelizable. In Fig. 4, we show the part of the local mesh that is modified, as it needs to be checked for validity and is updated in every step. The **normal move** does not change any of the mesh vertices. In order to generate the move, we create a vector $\boldsymbol{d}$ from a Gaussian distribution

$$d_i = \zeta; P(\zeta) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\zeta^2}{2\sigma^2}}, \tag{13}$$

where the components d_i are set to separate Gaussian random variables. The pseudorandom variables ζ are generated using a Mersenne-Twister algorithm, initialized by a time-dependent

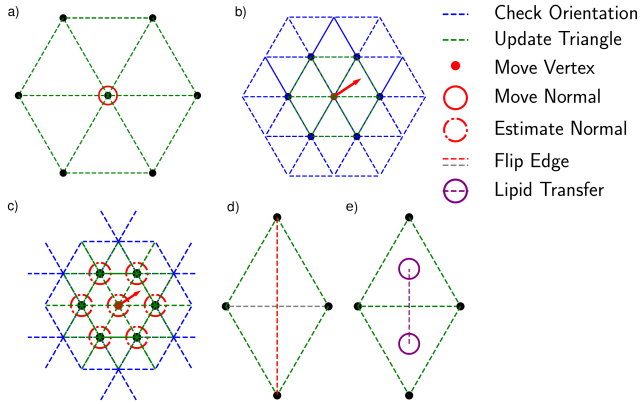

Figure 4: Implemented MC-moves: a) Normal move and its update range, b) vertex move and it is update and face orientation check range, c) Deep vertex move, d) Alexander move, e) Lipid transfer move. The legend describes the update schemes, face updates and orientation checks.

seed. The proposed new normal vector is then obtained as

$$\boldsymbol{n}_P = \frac{\boldsymbol{n} + \boldsymbol{d}}{\|\boldsymbol{n} + \boldsymbol{d}\|}. \tag{14}$$

Note that the probability of proposing $-\boldsymbol{d}$ is the same as the probability of proposing $\boldsymbol{d}$, due to the symmetry of the Gaussian distribution. The variance, σ, can be adjusted automatically. For proposal probability, we find

$$P(\boldsymbol{n}_P|\boldsymbol{n}) = P(\boldsymbol{n}|\boldsymbol{n}_P), \tag{15}$$

which helps to ensure detailed balance and allows us to compute the acceptance probability as

$$A(\boldsymbol{n}_P|\boldsymbol{n}) = \min\left\{1, e^{-\beta\Delta F}\right\}, \tag{16}$$

where ΔF is the change in the total free energy associated with the move and $\beta = (kT)^{-1}$. We state F in units of kT, cancelling β. The change in energy can be computed locally by considering only the triangles which contain this normal, hence it is trivially parallelizable. The only difficulty pertains to the evaluation of changes to the global quantities V and A. For a global "restraint" on a function Φ to Φ_0, with an energy contribution of the form

$$\frac{\lambda}{2}(\Phi - \Phi_0)^2 = F_\Phi. \tag{17}$$

The change $\Delta\Phi$ due to a local MC step leads to an energy change of

$$\Delta F_\Phi = \frac{\lambda}{2}\Delta\Phi(\Delta\Phi - 2(\Phi - \Phi_0)). \tag{18}$$

This means, that a local scheme requires the storage of the current global value Φ. However, Φ can be updated using $\Delta\Phi$ if the current local contribution to Φ is stored on the face.

In order to accept the proposed normal, it will also have to produce valid cubic or quadratic face interpolants. To ensure this, we require the proposed normal to be at a minimum angle to the other triangle normals $\boldsymbol{n}_i$, so that

$$\boldsymbol{n}_p \cdot \boldsymbol{n}_i > \epsilon = 0.01. \tag{19}$$

Finally, a collision check is performed. This collision check is delayed because of the great computation expense of evaluating the neighbor list. The second move to consider is the **vertex move**. Again, a vector is sampled from a normal distribution and added to the previous position. The same sort of symmetry argument, leading to Eq. (15) applies. The change of a vertex has more spatially extended consequences than the change of a normal, see Fig. 4, b). Of course, the triangles in the immediate vicinity of the initial vertex have to be updated. The acceptance criterion in Eq. (16) is applied. Moreover, the acceptance of the vertex move requires

$$\sphericalangle(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) > \frac{\pi}{16}, \tag{20}$$

for all of the angles in the triangle. The "**deep-vertex**" move from Fig. 4c, removes the normal vectors as independent degrees of freedom, determining them by the faces of the underlying triangular mesh. The way this is achieved is by reconstructing all normals affected by a vertex move, i.e. all normals on vertices neighboring the modified vertex are re-estimated using our normal estimation algorithm. Under these conditions, the move is fully reversible. This would not necessarily be true if the move was to only update the normal on its own vertex, because under these conditions the vertex normals of the surrounding faces would no longer correspond to their estimated values. As a remedy, we "quench" the normals to their estimated values before using this move or to use it standalone. Because of the potential issues, this move has been deactivated by default. Our MC scheme is also set-up in such a way as to minimize irreversibility by the organization of the Markov chain. In particular, for each degree of freedom, multiple trials(37) are proposed until acceptance, so that for each MC step (almost) all degrees of freedom have moved. The total step can, therefore, be interpreted as a factorization of individual symmetric steps.(38) Furthermore, we have implemented a mechanism of automated propagation, which ensures all other energy carrying objects are updated together with the faces, such object can e.g. include edge terms. Algorithmic details on the MC scheme and the automatic construction of decoupled parallel execution loads are given in the Supplementary Material. The **"Alexander"/bond-flip move** show in Fig. 4 d) is currently set to be accepted only if it decreases energy. Our automatic parallelization scheme, described in the SI relies on tracking the dependencies of each degree of freedom as determined from mesh. As the move changes the mesh topology, the parallelization structure of the code needs to be updated, and neighbor lists rebuilt after each round of bond-flipping moves. Hence, the remeshing move is executed only occasionally and (mostly) serially. As the number of faces is conserved, face properties (such as local bending rigidity) can remain associated with a fixed face, currently, no redistribution occurs. The Alexander move is associated with the edge which is flipped. The **lipid mixing** move shown in Fig. 4 e) is also edge-associated, random lipids are chosen from the population of both faces and transferred to another face. More information is found in the lipid mixing section.

Mesh Properties and Constraints

Constraints

It is possible to selectively block or remove parts of the mesh from active evolution. The former can be accomplished by removing some properties while retaining the mesh points. The latter is accomplished via *block* objects, which prevent updating parts of the mesh, e.g. only the x coordinate of a vertex or a normal. Blocks provide a simple way of implementing sophisticated boundary conditions. As an example, it is possible to allow the boundary of the mesh to evolve in-plane only while the normal vectors, and hence the tangent basis is kept constant. The block objects can also be used to suppress remeshing and lipid flow across membrane edges. We provide a utility to load simple coordinate and normal freezes via text files. The final distribution of our software contains a manual and selection tools to assist with preparing the blocks. By editing the source code, block objects can also be used to constrain curvilinear coordinates.

Face Properties

As energy and integral evaluations are performed over the faces, it makes sense to also store information about the parameters κ, J_s, A_0 and others on them. In fact, each face contains an arbitrary collection of properties, which are accessed via a text key. Such properties can be assigned at load as well as at run time and are automatically included in output visualizations. This allows us to dynamically store and export information about composition as well as to compute modified elastic properties and local coupling terms on the fly.

Lipid Mixing

The implemented energy functionals would also allow to compute the energy directly for a composition. However, a proper implementation of lipid mixing requires the lipid composition to be dynamic and, therefore, requires the definition of mixing rules and moves which manipulate it. Ideally, the modification of the lipid composition should be transparent to most of the energies computed from it. In the literature, it has often been proposed to compute the bending rigidity κ as(39, 40) via a harmonic average of the single lipid moduli κ_i:

$$\frac{1}{\kappa} = \sum_i \phi_i \frac{1}{\kappa_i}. \tag{21}$$

Here ϕ_i refers to the volume fraction, which is calculated as

$$\phi_i = \frac{m_i A_i}{\sum_j m_j A_j}, \tag{22}$$

with A_i the area per lipid of the lipid species i. This means, that we assume that the area per lipid does not change while mixing. The area per lipid A_i is a reference area contributing to A_0, so that mixing/bending and stretching are not coupled. By m_i, we refer to the number of lipids of species i present in a triangle. Of course, completely different behavior is possible. An example occurs during a phase transition to gel or L_O phase. As long as the behavior is well-defined it is easy to implement the requisite models in our code. The spontaneous curvature is computed as

$$J_s = \sum_i \phi_i c^0_{s,i}, \tag{23}$$

where c^0_i is spontaneous curvature of the pure i.(13, 41) We have recently tested the validity of these equations against atomistic molecular dynamics results.(42) Furthermore, the mixing entropy of the lipid composition with a total occupancy of lipids M on a side/leaflet of a triangle can be estimated as ideal in the surface fractions(41):

$$F_{\text{mix}} = kTM \sum_i \phi_i \ln\phi_i. \tag{24}$$

This free energy is added to the energy in the form of a penalty. In this way, it can be used with all energy functionals. Real valued molar fractions ϕ_i and lipid occupancies m_i presume that the lipid composition of a triangle is a macroscopic quantity. As the mixing entropy is temperature dependent, the temperature is provided via the inverse temperature β. Furthermore, both leaflets in a bilayer are considered, and their properties are computed separately, and then added together. The curvature c_0 is added with an inverse sign to that of the opposite leaflet. The equilibrium area of a face A_0 is computed from the areas per lipid on each leaflet, the A_0 are then averaged (we recommend setting up the simulation with equal A_0 for both leaflets). The implemented approach models only the bilayer midplane, so that the finite and variable thickness of the actual bilayer is neglected. There is also the possibility to use a stylized protein, whose move will be integer valued and which will override the curvature generated by the lipids. In order to simulate lipid mixing, an exchange move has been implemented. On each leaflet:

1. Select two lipid types at random from adjacent faces.
2. Generate two random amounts of lipids to be moved.
3. Move the two lipids between the adjacent faces and recompute the energies, including mixing energy.
4. Accept or reject the move.

Uniform random numbers are used for the exchange move. In order for the move to be allowed, the total occupancy must be above a certain threshold, e.g. zero. Currently, the moves are set so that the total A_0 of each face and leaflet does not change by lipid exchange flow. This serves to prevent mesh degeneration. While we are not aware of an equivalent implementation, similar ideas have been explored in the past.(43)

Collision Detection

In a naive implementation, the detection of potential collisions between any pair of elements in a set of n triangles requires $O(n^2)$ operations. To mitigate the computational expense of the collision detection, we implemented a neighbor-list based *broad* (44) search that aims to reduce the number of explicit intersections which have to be computed. The broad search is followed by a *narrow* phase that relies on the popular (45–47) fast triangle-triangle intersection test of Möller (48). As the mesh faces are cubic surfaces, the use of the Möller algorithm is a compromise between precision and computational cost.

Penalties and Errors

For the calculation of the energy, and the consistency of the interpolation, we require a consistent definition of the surface integral everywhere on the surface and to evaluate the same for an arbitrary subdomain

$$\int_{\text{Patch}} f \, \mathrm{d}A, \tag{25}$$

where f may depend on derivatives of the surface or might be a differential form. For example, a smooth surface in $\mathbb{R}^3$ will fulfill

$$\int_A \nabla \times \hat{\boldsymbol{n}} \cdot \mathrm{d}\boldsymbol{A} = \oint_{\partial A} \hat{\boldsymbol{n}} \cdot \mathrm{d}\boldsymbol{l} = 0, \tag{26}$$

for any closed, finite area A with boundary ∂A, tangent vector $\mathrm{d}\boldsymbol{l}$ and oriented surface element $\mathrm{d}\boldsymbol{A}$, and . It is easily shown that $\nabla \times \hat{\boldsymbol{n}} = 0$ is fulfilled everywhere inside an interpolated patch. If $\mathrm{d}\boldsymbol{l}$ is a tangent vector of an edge interpolant, integrating around the boundary of each patch will also fulfill the condition as even where normals are ambiguous/discontinuous on the edge interpolant, both are orthogonal to the tangent vector which follows the boundary line by construction. However, we need more regularity than this, as we are interested in integrals of $\tilde{H}$ and $\tilde{H}^2$. On a patch boundary, one of the tangent vectors, i.e. the one taken along the edge interpolant is identical to that of its neighboring patch. Away from the vertex, the other tangent vector may be different so that for an identical point at the edge two different tangent vectors are obtained. *Quasi*-tangent continuity means that tangent continuity is ensured only at the vertices so that in general, our interpolant is only C^0 continuous, across edges but the computation of curvature requires derivatives of $\boldsymbol{n}$ in any direction, in particular the total curvature $\tilde{H}$, i.e. the covariant normal divergence

$$\tilde{H} = \nabla \cdot \hat{\boldsymbol{n}}, \tag{27}$$

has to be (square) integrable on the surface. At the edge, one tangent vector is shared (hence the same normal curvature in one direction(49)). If the normal vector at this point is also identical, it is possible to construct a local orthogonal tangent basis, this second tangent ($\boldsymbol{t}_\perp$) defines a shared normal $\hat{\boldsymbol{n}}$ as it stands orthogonal to the edge direction and $\boldsymbol{n}$ at all times. Tangent, i.e. C^1 continuity is sufficient to guarantee some curvature integral properties, but does not guarantee continuity of the curvature itself. Furthermore, numerical experiments showed that the patch interpolant does not always reproduce the vertex normals used to create it. For example, one of the edge interpolants may have parallel tangents at the endpoints during construction or might generate normal vectors of opposite orientation. Hence, before accepting a move it is verified that

$$\hat{\boldsymbol{n}}_{\text{Mesh}}\hat{\boldsymbol{n}}_{\text{Nagata}} \geq 1 - \epsilon. \tag{28}$$

We use $\epsilon = 0.005$ as numerical tolerance for normal reproduction. In Fig. 4 of the Supplementary Material, we show how enforcing normal reproduction helps to enhance tangent continuity. In order to bring the discretization closer to a C^1 continuity, we define the following energy scale

$$I = [(\kappa + 1)(1 + A_0 J_s^2) + A_0^2 K_A]. \tag{29}$$

The individual terms in this intensity are spatial-scale free energy units. They are set to cover all sources of energy in the problem so that a significant J_s does not damage the regularizing penalty. Now, we split the tangent basis into two parts,

$$\boldsymbol{t}_= = \boldsymbol{\varphi}'; \boldsymbol{t}_\perp = \boldsymbol{\varphi}' \times \hat{\boldsymbol{n}}. \tag{30}$$

We then require that at the midpoint of an edge

$$\hat{\boldsymbol{t}}_{\perp,m} \stackrel{!}{=} \frac{1}{2}(\hat{\boldsymbol{t}}_{\perp,0} + \hat{\boldsymbol{t}}_{\perp,1}), \tag{31}$$

where the subscripts (0,1) denote the normalized average value of $\boldsymbol{t}_\perp$ at the endpoints of the edge interpolants. By controlling the evolution of $\hat{\boldsymbol{t}}_\perp$, in each triangle, we enforce the continuity of normals at the center of the triangle, biasing towards C^1 continuity without requiring information from adjacent triangles. Deviation from Eq. 31 carries a quadratic penalty depending on I. In the Supplementary Material, we quantify the approximate tangent continuity and show its conservation in a production run. We also penalize too small triangle areas (starting $< 40\%A_0$) and deviations from an equilibrium angle ($|\alpha| < 0.316$) with quadratic penalties proportional to I. The interpolant is highly tolerant of different penalties, as long as they are sufficiently strong. The area penalty serves mainly to prevent mesh degeneration. This mesh degeneration is less of a problem for the interpolant than for the MC algorithm, as the admissible step size becomes highly in-homogeneous. Similarly, the angle penalty mostly serves to prevent the locking and overlap of triangles. The numerical values of the penalty terms are given in the Supplementary Material.

Input and Output Formats

The mesh input file format for this code is wavefront OBJ. Meshes of this format can be generated and exported from many free and standard tools. If the original obj file does not contain normals, they will be reconstructed from neighboring face normals in a simple weighting scheme(50). The program also exports this type of mesh, in addition to XML-VTK(51) unstructured grid files (.vtu). These are readily available and can be visualized and processed with the Paraview software package(52).

RESULTS AND DISCUSSION

Example Meshes

It is difficult to unit-test the error of the interpolation. The easiest way to proceed is to demonstrate the accuracy for example meshes. These example meshes are not the result of a simulation, but are generated geometric objects. The results are shown in Fig. 5. For the unit icosphere mesh Fig. 5 a), the corresponding values are a total curvature of 12.518, energy of 24.959κ and area of 12.5662. The Gaussian curvature integral evaluates to 12.4707. The theoretical values for area, curvature and Gaussian curvature are $4\pi \approx 12.5664$ for $R = 1$. The theoretical energy is $8\pi\kappa$. The volume error is negligible. We also tested the icosphere using purely cubic interpolants. Area and volume accuracy remained unaffected, but energy decreased to 24.736κ and curvature to 12.44. For the catenoid the absolute error was similarly increased by 0.24κ. We note, that the quadratic interpolation appears more robust, but errors remain small. At present, we do not know how important integration order is for error control. We fall back on the quadratic interpolant, when normals allow it. The toroidal mesh Fig. 5 b) has a total K integral of -0.2 over the surface, the theoretical value is zero. The catenoid mesh Fig. 5 c), mainly due to the difficult normal estimation at the edges, results in a total curvature integral of -0.27 over all faces and a total Helfrich energy of 0.22κ over an area of 17.63 units. Any minimal surface has $H = 0$ everywhere, so the theoretical value for mean curvature integrals is zero. The catenoid also has an Euler characteristic of 0. As such, integrating K over the mesh and k_G over the edges results in two terms canceling to 0.032, close to the expected result.. This illustrates the high accuracy of our scheme for good quality meshes. The flat triangle mesh Fig. 5 d) does have a mean and Gaussian curvature identically zero. Its k_G boundary integral evaluates to 6.28, close to the theoretical value of 2π for the Euler index of one. In the lower panel, we show the convergence of Gaussian and mean curvature for the platonic solids. For their computation, we used analytical normals.

Helfrich Shapes

In order to validate the interpolant and our MC algorithms, we reproduced the behavior of the known rotationally symmetric shapes of the Helfrich Hamiltonian. The corresponding

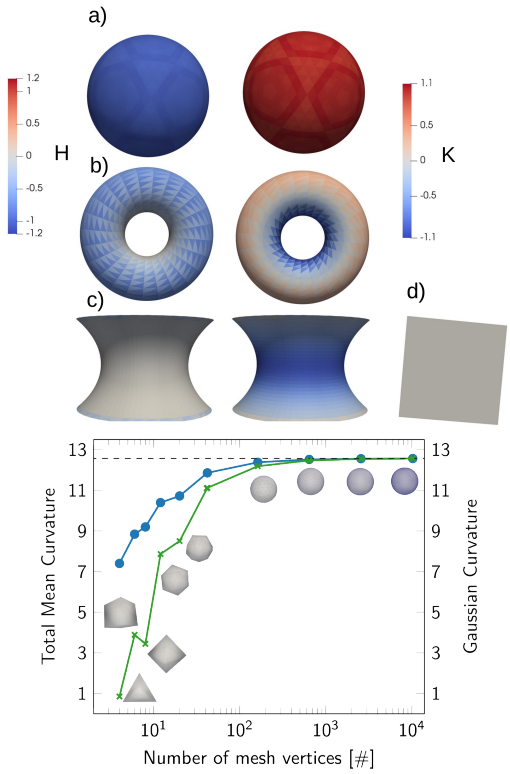

Figure 5: Upper part: Four example meshes, with mean curvatures (left column) and Gaussian curvatures (right column) plotted as averages over face integrals, normal orientation gives negative H to the sphere. a) Icosphere mesh (642 vertices, 1280 faces) b) Toroidal mesh (1152 faces, 576 vertices), R_1=1, R_2 = 2. c) Catenoid Mesh. (2340 faces, 1209 vertices) Flat mesh (162 faces, 100 vertices. The mean curvature and Gaussian curvature scale are shared between meshes. Lower part: Convergence test of mean (blue) and Gaussian curvature (green) integrals for a range of platonic solids and icospheres.

minima have been well explored and confirmed recently by independent techniques.(25, 30) Results are shown in Fig. 6. We have generated all the points using a discocyte starting geometry, by moving the volume restraint and keeping the area constant. In particular, thanks to the MC implementation we are able to transform prolate to discocyte and vice-versa near the theoretical limits, which are indicated by the dotted vertical lines with the numerical values marked in Fig 6.(25, 30) The prolate state can be somewhat metastable, so that, with fast pulling some "wormlike" shapes were obtained. Discoid geometries readily transition to prolate ones: the point to on right of the line at v_0 0.65 has a prolate long axis. The transition from discocyte to stomatocyte was also observed and occurred via collision and repulsion of the stomatocyte dimples. We provide documentation of these transitions as videos, showing how we can generate each shape from each other. The values we show for $v_0 > 0.1$, represent sampling averages over 20K steps, after more than 1M equilibration steps at the stationary shape as we have not implemented gradients. In theory, the minimal energy is $8\pi\kappa$ for the sphere at $v_0 = 1$, and $16\pi\kappa$ for the stomatocyte (in the limit a sphere inside as sphere), with a supralinear increase in the prolate regime.(35) The theoretical curve is shifted by the thermal energy α. The value of α was estimated by comparing the minimal curvature energy of an icosphere at $v_0 = 1$ at rest, found when annealing only the normals and edge flipping at $\beta = 1000$ to its simulation average energy at the simulation conditions, the difference is $\approx 0.7 \cdot 8\pi\kappa$. The geometries shown in Fig. 6 also show of the curvature on each individual face (only the average values are mapped on each face).

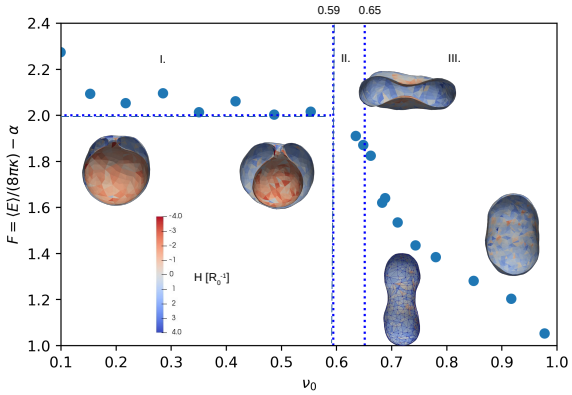

Figure 6: Results for spherical vesicles, including the shapes for stomatocytes I., discocytes and oblate II., prolate III. shapes obtained using volume and area constraints. The blue line indicates the theoretical value for the stomatocyte. α is due to the thermal energy of the degrees of freedom of the mesh. All results are derived from the 1280 face / 642 icosphere mesh. Starting point was a discocyte geometry. The energies are average values over the equilibrated part of the run.

For this simulation, only the global surface area A and the global volume V were restrained. From these was computed the reduced volume

$$v_0 = \frac{6V}{A^{3/2}}\sqrt{\pi}. \tag{32}$$

When the space inside the vesicle becomes too small, i.e. at $v_0 < 0.2$ fluctuations deform the mesh and the neck becomes heavily penalized. While we believe it might be possible to go even lower with appropriate penalties, this seems to be quite sufficient for most stomatocyte shapes. In the Supporting Material we provide further simulation details. Equivalent inputs are also part of the OrganL distribution.

Examples: Curvature Bias and Lipid Mixing

The sampling of the standard Helfrich minima in Fig. 6 is just a precondition for the more advanced methods. With a robust description of the surface, it also becomes possible

to add the lipid moves described in the previous section. For the purposes of clarity, we have chosen only one leaflet to have a strong curvature. In Fig. 7 a), the comparison of a mixed patch with $J_s = 0$ is shown. A protrusion is stabilized by curvature-driven lipid demixing. The lipid population and local J_s are visualized together in central and right panels. The boundary of the simulation is constrained. The lipid demixing effects in the example are intended for illustration purposes, they result from the boundary conditions, lipid parameters and restraints applied, and do not indicate that ideal mixtures will spontaneously demix to create deformations under realistic conditions.

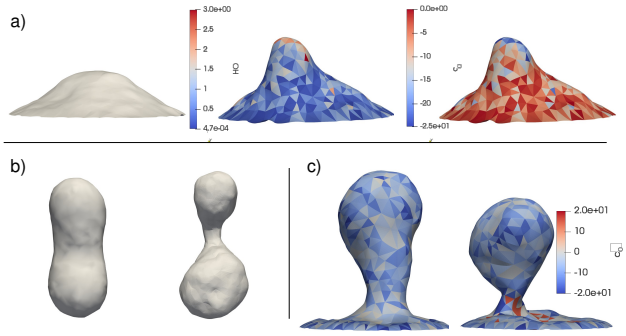

Figure 7: a) Coupling of lipid mixing and curvature generation with membrane geometry. Left panel: Simulation of lipid mixing without curvature, Central panel distribution of curved lipid and budding triggered by the same. Right panel, spontaneous curvature. b) ADE simulation starting from prolate geometry (left), as implemented by a global curvature restraint. c) Comparison of lipid budding and constriction with high spontaneous curvature (left), and neck-constricting protein mimicks (right). All spherical cap simulations had 968 faces and 509 vertices, the ADE simulation used a prolate starting geometry (642 vertices, 1280 faces).

Apart from a restriction on volume and surface integrals, it is also possible to restrict the mean curvature integral. This is equivalent to the ADE formulation, as area differences create global curvature restrictions. One example of such an ADE simulation is shown in Fig. 7 b). The left side is the starting geometry and only the curvature restriction enforces the panhandle shape. Finally, it is possible to slightly relax the penalties to bring forth more flexibility. Under these conditions, it is possible to create and constrict multiple buds from a small spherical cap and generate deformations on buddings, etc. One example is shown in Fig. 7 c), which points towards the possibilities that can be reached by the refinement of moves and the introduction of explicit proteins. Example input files are distributed with the software. We also provide for the possibility to explicitly work with Gaussian curvature, line tension and open meshes, but these features are untested with evolving meshes.

CONCLUSION

We described a method for the mesoscopic modeling of lipid membranes and related systems, based on a powerful but simple discretization scheme. This scheme has been introduced here to address previous weaknesses of the Nagata interpolant and provides a new approach to DTS simulations. What distinguishes our method is the possibility to assign extra structure, such as differentiable functions to the faces and edges. Our method also provides a direct correspondence between analytical formulation of the curvature elastic theory and its implementation. Consequently, we also introduce a feature-rich implementation in the form of a Monte Carlo code. The software was shown to be capable of modeling the well-known rotational equilibrium shapes of the Helfrich energy, at least semi-quantitatively and was empirically found to provide correct relative energies. In its present form, the method introduced here is potentially useful for predicting e.g. the effect of lipid compositions, boundaries and proteins in a number of stylized ways. Our software, OrganL, can be used to study viral budding, curvature sorting and cellular shapes, membrane shape fluctuations and many other effects. The implementation design allows for easy extension of the code to include new couplings, adding e.g. a cytoskeleton. As an example we included edge element-dependent energy contributions for open meshes. The current limitations are the lack of explicit surface-surface interactions beyond collisions (such as protein membrane interactions), gradient, for explicit forces for fluid-structure interaction and topological changes, such as membrane fusion.

AUTHOR CONTRIBUTIONS

CA designed the research and wrote the code. BF contributed to the collision detection. CA and BF carried out all simulations and analyzed the data. CA and MD wrote the article, with input and edits from BF.

DECLARATION OF INTERESTS

The authors declare no competing interests.

ACKNOWLEDGMENTS

CA and MD thank Charles University for support via the PRIMUS research project PRIMUS/20/SCI/015. The authors also thank Vojtěch Kubáč for his contributions regarding the visualization and informative early work, Petr Pelech and Hina Arif for proofreading and Jovi K for tests and cross-plattform support. CA has been supported by Charles University Research Centre program No. UNCE/24/SCI/005.

REFERENCES

1. Noid, W. G., 2023. Perspective: Advances, Challenges, and Insight for Predictive Coarse-Grained Models. *J. Phys. Chem. B* 127:4174–4207.

2. Evans, E., 1974. Bending resistance and chemically induced moments in membrane bilayers. *Biophys. J.* 14:923–931.

3. Canham, P., 1970. The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell. *J. Theor. Biol.* 26:61–81.

4. Helfrich, W., 1973. Elastic Properties of Lipid Bilayers: Theory and Possible Experiments. *Z. Naturforsch. C* 28:693–793.

5. Golani, G., N. Ariotti, R. G. Parton, and M. M. Kozlov, 2019. Membrane curvature and tension control the formation and collapse of caveolar superstructures. *Dev. Cell* 48:523–538.

6. Bassereau, P., R. Jin, T. Baumgart, M. Deserno, R. Dimova, V. A. Frolov, P. V. Bashkirov, H. Grubmüller, R. Jahn, H. J. Risselada, L. Johannes, M. M. Kozlov, R. Lipowsky, T. J. Pucadyil, W. F. Zeno, J. C. Stachowiak, D. Stamou, A. Breuer, L. Lauritsen, C. Simon, C. Sykes, G. A. Voth, and T. R. Weikl, 2018. The 2018 biomembrane curvature and remodeling roadmap. *J. Phys. D: Appl. Phys.* 51:343001.

7. Manor, U., S. Bartholomew, G. Golani, E. Christenson, M. Kozlov, H. Higgs, J. Spudich, and J. Lippincott-Schwartz, 2015. A mitochondria-anchored isoform of the actin-nucleating spire protein regulates mitochondrial division. *eLife* 4:e08828.

8. Pezeshkian, W., M. König, T. A. Wassenaar, and S. J. Marrink, 2020. Backmapping triangulated surfaces to coarse-grained membrane models. *Nat. Commun.* 11:2296.

9. Fedosov, D. A., W. Pan, B. Caswell, G. Gompper, and G. E. Karniadakis, 2011. Predicting human blood viscosity in silico. *Proc. Natl. Acad. Sci. USA* 108:11772–11777.

10. Allolio, C., A. Haluts, and D. Harries, 2018. A Local Instantaneous Surface Method for Extracting Membrane Elastic Moduli from Simulation: Comparison with other Strategies. *Chem. Phys.* 514:31 – 43.

11. Allolio, C., and D. Harries, 2021. Calcium Ions Promote Membrane Fusion by Forming Negative-Curvature Inducing Clusters on Specific Anionic Lipids. *ACS Nano* 15:12880–12887.

12. Watson, M. C., E. G. Brandt, P. M. Welch, and F. L. H. Brown, 2012. Determining Biomembrane Bending Rigidities from Simulations of Modest Size. *Phys. Rev. Lett.* 109:028102.

13. Khelashvili, G., D. Harries, and H. Weinstein, 2009. Modeling Membrane Deformations and Lipid Demixing upon Protein-Membrane Interaction: The {BAR} Dimer Adsorption. *Biophys. J.* 97:1626 – 1635.

14. Hu, M., P. Diggins, and M. Deserno, 2013. Determining the bending modulus of a lipid membrane by simulating buckling. *J. Chem. Phys.* 138:214110.

15. G. Gompper, and D.M. Kroll, 1996. Random Surface Discretizations and the Renormalization of the Bending Rigidity. *J. Phys. I France* 6:1305–1320.

16. Gompper, G., and D. M. Kroll, 1998. Membranes with Fluctuating Topology: Monte Carlo Simulations. *Phys. Rev. Lett.* 81:2284–2287.

17. Gompper, G., and D. M. Kroll, 1997. Network models of fluid, hexatic and polymerized membranes. *J. Phys.: Condens. Matter* 9:8795.

18. Frank Jülicher, 1996. The Morphology of Vesicles of Higher Topological Genus: Conformal Degeneracy and Conformal Modes. *J. Phys. II France* 6:1797–1824.

19. Kantor, Y., and D. R. Nelson, 1987. Phase transitions in flexible polymeric surfaces. *Phys. Rev. A* 36:4020–4032.

20. Meyer, M., M. Desbrun, P. Schröder, and A. H. Barr, 2003. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *In* H.-C. Hege, and K. Polthier, editors, Visualization and Mathematics III. Springer Berlin Heidelberg, Berlin, Heidelberg, 35–57.

21. McWhirter, J. L., H. Noguchi, and G. Gompper, 2009. Flow-induced clustering and alignment of vesicles and red blood cells in microcapillaries. *Proc. Natl. Acad. Sci. USA* 106:6039–6043.

22. Tamemoto, N., and H. Noguchi, 2021. Reaction-diffusion waves coupled with membrane curvature. *Soft Matter* 17:6589–6596.

23. Vutukuri, H. R., M. Hoore, C. Abaurrea-Velasco, L. van Buren, A. Dutto, T. Auth, D. A. Fedosov, G. Gompper, and J. Vermant, 2020. Active particles induce large shape deformations in giant lipid vesicles. *Nature* 586:52–56.

24. Ramakrishnan, N., P. Sunil Kumar, and R. Radhakrishnan, 2014. Mesoscale computational studies of membrane bilayer remodeling by curvature-inducing proteins. *Phys. Rep.* 543:1–60. Mesoscale computational studies of membrane bilayer remodeling by curvature-inducing proteins.

25. Bian, X., S. Litvinov, and P. Koumoutsakos, 2020. Bending models of lipid bilayer membranes: Spontaneous curvature and area-difference elasticity. *Comput. Methods Appl. Mech. Eng.* 359:112758.

26. Siggel, M., S. Kehl, K. Reuter, J. Köfinger, and G. Hummer, 2022. TriMem: A parallelized hybrid Monte Carlo software for efficient simulations of lipid membranes. *J. Chem. Phys.* 157.

27. Sachin Krishnan, T. V., S. L. Das, and P. B. S. Kumar, 2019. Transition from curvature sensing to generation in a vesicle driven by protein binding strength and membrane tension. *Soft Matter* 15:2071–2080.

28. Brakke, K. A., 1992. The Surface Evolver. *Exp. Math.* 1:141–165.

29. Pezeshkian, W., and J. H. Ipsen, 2019. Fluctuations and conformational stability of a membrane patch with curvature inducing inclusions. *Soft Matter* 15:9974–9981.

30. Chen, J., T. Yu, P. Brogan, R. Kusner, Y. Yang, and A. Zigerelli, 2021. Numerical methods for biomembranes: Conforming subdivision methods versus non-conforming PL methods. *Math. Comp.* 90:471–516.

31. Barrett, J., H. Garcke, and R. Nürnberg, 2017. Finite element approximation for the dynamics of asymmetric fluidic biomembranes. *Math. Comput.* 86:1037–1069.

32. Nagata, T., 2005. Simple local interpolation of surfaces using normal vectors. *Comput. Aided Geom. Des.* 22:327–347.

33. Neto, D., M. Oliveira, L. Menezes, and J. Alves, 2013. Improving Nagata patch interpolation applied for tool surface description in sheet metal forming simulation. *Comput. Aided Des.* 45:639–656.

34. Laursen, M. E., and M. Gellert, 1978. Some criteria for numerically integrated matrices and quadrature formulas for triangles. *Int. J. Numer. Method. Eng.* 12:67–76.

35. Seifert, U., K. Berndl, and R. Lipowsky, 1991. Shape transformations of vesicles: Phase diagram for spontaneous-curvature and bilayer-coupling models. *Phys. Rev. A* 44:1182–1202.

36. Miao, L., U. Seifert, M. Wortis, and H.-G. Döbereiner, 1994. Budding transitions of fluid-bilayer vesicles: The effect of area-difference elasticity. *Phys. Rev. E* 49:5389–5407.

37. Liu, J. S., F. Liang, and W. H. Wong, 2000. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *J. Am. Stat. Assoc.* 95:121–134.

38. Krauth, W., 2021. Event-Chain Monte Carlo: Foundations, Applications, and Prospects. *Front. Phys.* 9:663457.

39. Khelashvili, G., B. Kollmitzer, P. Heftberger, G. Pabst, and D. Harries, 2013. Calculating the Bending Modulus for Multicomponent Lipid Membranes in Different Thermodynamic Phases. *J. Chem. Theory Comput.* 9:3866–3871.

40. Kozlov, M. M., and W. Helfrich, 1992. Effects of a cosurfactant on the stretching and bending elasticities of a surfactant monolayer. *Langmuir* 8:2792–2797.

41. Andelman, D., M. M. Kozlov, and W. Helfrich, 1994. Phase Transitions between Vesicles and Micelles Driven by Competing Curvatures. *Europhys. Lett.* 25:231.

42. Konar, S., H. Arif, and C. Allolio, 2023. Mitochondrial Membrane Model: Lipids, Elastic Properties and the Changing Curvature of Cardiolipin. *Biophys. J.* 122:4274–4287.

43. Sunil Kumar, P. B., G. Gompper, and R. Lipowsky, 2001. Budding Dynamics of Multicomponent Membranes. *Phys. Rev. Lett.* 86:3911–3914.

44. Ericson, C., 2004. Real-time collision detection. Crc Press, Boca Raton.

45. Kurniawan, R., and T. J. Ko, 2019. Surface topography analysis in three-dimensional elliptical vibration texturing (3D-EVT). *Int. J. Adv. Manuf. Technol.* 102:1601–1621.

46. Landstorfer, M., B. Prifling, and V. Schmidt, 2021. Mesh generation for periodic 3D microstructure models and computation of effective properties. *J. of Computat. Phys.* 431:110071.

47. Mandal, A., P. Chaudhuri, and S. Chaudhuri, 2022. Interactive Physics-Based Virtual Sculpting with Haptic Feedback. *Proc. ACM Comput. Graph. Interact. Tech* 5:1–20.

48. Möller, T., 1997. A fast triangle-triangle intersection test. *J. Graph. Tools* 2:25–30.

49. do Carmo, M., 2016. Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition. Dover Books on Mathematics. Dover Publications.

50. Max, N., 1999. Weights for Computing Vertex Normals from Facet Normals. *J. Graph. Tools* 4:1–6.

51. Kitware Inc., 2010. VTK User's Guide. Kitware Inc., Clifton Park.

52. Kitware Inc., 2022. ParaView 5.11. www.paraview.org.

SUPPLEMENTARY MATERIAL

The Supplementary Material contains a detailed derivation of the interpolants, algorithmic details about collision detection, quadrature and parallel execution as well as data structures, penalty constants and other numerical parameters (such as simulation run parameters) and a description of additional energy functionals. It also contains more information about controlling tangent continuity. The program source code, supplementary videos and manual are available from the corresponding author or under https://github.com/allolio/organl. The program is free of charge for non-commercial use.