The Journal of Chemical Physics | ARTICLE | scitation.org/journal/jcp

TriMem: A parallelized hybrid Monte Carlo software for efficient simulations of lipid membranes

Cite as: J. Chem. Phys. **157**, 174801 (2022); doi: 10.1063/5.0101118
Submitted: 29 May 2022 • Accepted: 30 September 2022 •
Published Online: 1 November 2022

View Online | Export Citation | CrossMark

Marc Siggel,[1,a] Sebastian Kehl,[2] Klaus Reuter,[2] Jürgen Köfinger,[1] and Gerhard Hummer[1,3,b]

AFFILIATIONS

[1] Department of Theoretical Biophysics, Max Planck Institute of Biophysics, Max-von-Laue-Straße 3, 60438 Frankfurt am Main, Germany
[2] Max Planck Computing and Data Facility, Gießenbachstraße 2, 85748 Garching, Germany
[3] Institute of Biophysics, Goethe University Frankfurt, 60438 Frankfurt am Main, Germany

[a] **Current addresses:** European Molecular Biology Laboratory Hamburg, Notkestraße 85, 22607 Hamburg, Germany and Centre for Structural Systems Biology (CSSB), Notkestraße 85, 22607 Hamburg, Germany.
Electronic mail: marc.siggel@embl-hamburg.de
[b] **Author to whom correspondence should be addressed:** gerhard.hummer@biophys.mpg.de

ABSTRACT

Lipid membranes are integral building blocks of living cells and perform a multitude of biological functions. Currently, molecular simulations of cellular-scale membrane remodeling processes at atomic resolution are extremely difficult, due to their size, complexity, and the large times-scales on which these processes occur. Instead, elastic membrane models are used to simulate membrane shapes and transitions between them and to infer their properties and functions. Unfortunately, an efficiently parallelized open-source simulation code to do so has been lacking. Here, we present TriMem, a parallel hybrid Monte Carlo simulation engine for triangulated lipid membranes. The kernels are efficiently coded in C++ and wrapped with Python for ease-of-use. The parallel implementation of the energy and gradient calculations and of Monte Carlo flip moves of edges in the triangulated membrane enable us to simulate large and highly curved membrane structures. For validation, we reproduce phase diagrams of vesicles with varying surface-to-volume ratios and area difference. We also compute the density of states to verify correct Boltzmann sampling. The software can be used to tackle a range of large-scale membrane remodeling processes as a step toward cell-scale simulations. Additionally, extensive documentation make the software accessible to the broad biophysics and computational cell biology communities.

I. INTRODUCTION

Living cells are bounded by lipid membranes, and the interior of eukaryotic cells is filled with membranous organelles. Cellular membrane structures are highly dynamic, strongly curved, and branched.[1,2] Membranes are flexible and behave like two-dimensional fluids. Super-resolution microscopy and cryo-electron tomography (cryo-ET) have given unprecedented insights into the spatial organization of cellular membranes and their associated protein structures.[3] However, a detailed and comprehensive biophysical understanding of the mechanisms underlying organellar membrane reshaping in the cellular context remains elusive.

Molecular dynamics simulations are increasingly becoming state of the art for understanding membrane biophysics in the cellular context.[4–7] Multi-scale approaches combining all-atom, coarse-grained and meso-scale methods are used to tackle various aspects of membrane remodeling.[8,9] All-atom models are used to gain detailed insight into protein–lipid interactions and protein–protein interactions inside membranes but are limited in size and time scale.[9] Coarse-grained particle-based simulations make it possible to study large membrane-associated complexes and complex membrane shape changes.[9–13] However, simulations of large cell-scale membrane remodeling processes are currently extremely difficult with molecular models at atomic or near-atomic resolution mainly due to system size and the time scales on which the processes occur. Computational resources limit the time and length scales that can be studied with particle-based approaches.[14] Therefore, large-scale membrane remodeling processes relevant to biological processes of organelles, vesicles, and cells are commonly studied by using meso-scale models and continuum approaches, primarily membrane-elastic theory.[15,16]

Dynamic triangulated surfaces (DTSs) have emerged as a useful meso-scale model to solve the Helfrich–Hamiltonian numerically and study large-scale membrane-shaping processes. Initially, DTSs were used to study shaping and properties of fluctuating giant unilamellar vesicles (GUVs).[17–20] The discretization of membranes makes it possible to sample non-axisymmetric shapes[19] beyond symmetric arc length parameterizations.[21,22] This method has been applied to a number of biological processes with increasing complexity, including nano-particle wrapping,[23,24] membrane tubulation,[25] formation of autophagic vesicles,[26] formation of Golgi stacks,[27] and protein-induced membrane budding.[28,29]

Here, we present TriMem, an open-source software package for efficient simulation and optimization of DTSs using a parallelized hybrid Monte Carlo (MC) approach. This approach overcomes limitations of existing serial implementations.[25,26,28,30,31] In such non-parallel approaches, the number of vertices of the triangulated membrane representation is severely limited to keep computational times manageable. Yet, large numbers of vertices are needed to describe highly curved membranes, which are ubiquitous in cellular organelles such as mitochondria or the tubular endoplasmic reticulum (ER). As a further challenge, the commonly used random single-particle MC moves are inefficient with respect to sampling compared to a molecular dynamics (MD)-based move, which has an acceptance rate of nearly one.[32] Importantly, only few code bases for these types of simulations have been made freely available to the broader community despite the large amount of scientific research in this area, and none of them are parallelized yet.[30,31]

This paper is structured as follows: First, we briefly recapitulate the discretized version of Helfrich theory and the Helfrich–Hamiltonian used throughout this work (Sec. II). We then introduce the TriMem software, including its implementation strategy and algorithms (Sec. III). We present benchmark results on timings for varying mesh sizes and compare the performance to a serial single-core implementation (Sec. III D). We validate the software by reproducing well established phase diagrams of vesicle shapes and test for Boltzmann sampling by calculating the density of states (Sec. IV). Finally, we provide a comprehensive outlook on the expected impact of our software and on future developments (Sec. VI).

II. MEMBRANE MODEL

Membrane-elastic theory describes membranes as 2D surfaces with fluid properties embedded in 3D space. In its simplest form, the bending free energy of a membrane can be written in terms of the so-called Helfrich–Hamiltonian[15,16]

$$H_B = \frac{\kappa_B}{2} \oint \mathrm{d}A\,(2H - C_0)^2 + \kappa_G \oint dA\,K_G, \tag{1}$$

with $H = 0.5(c_1 + c_2)$ the mean curvature and $K_G = c_1 c_2$ the Gaussian curvature, where c_1 and c_2 are the principal curvatures. C_0 signifies the spontaneous curvature of the membrane that is modulated by a variety of factors, including proteins, lipid composition, and membrane asymmetry. κ_B and κ_G are the bending rigidity and Gaussian bending modulus, respectively, and describe the elastic properties of the studied membrane. The second term, the Gaussian curvature, can usually be neglected because it is constant in the absence of changes of the topology according to the Gauss–Bonnet theorem. Multiple extensions and variations of the Helfrich–Hamiltonian have been introduced to include a broad spectrum of external factors, such as area difference, protein inclusions, and osmotic pressure. Variational minimization of the Helfrich–Hamiltonian with respect to the membrane shape leads to fourth-order differential equations, which have been solved analytically only for a limited number of cases of high symmetry.[33,34] However, numerical solutions are achievable.

To make simulations of membrane shapes computationally tractable and amenable to integration and MC sampling, the surface is discretized as a triangular mesh, which can, in principle, represent arbitrary closed surfaces. The Hamiltonian of this discretized system (H_{tot}) is more complex than the original Helfrich–Hamiltonian. We decompose the energy as

$$H_{\mathrm{tot}} = E_B + E_V + E_A + E_{\Delta A} + E_T + E_R, \tag{2}$$

where E_B is the bending energy, E_V the volume energy to maintain the total internal volume, E_A the area energy to maintain the total surface area, $E_{\Delta A}$ the area-difference energy (ADE), E_T the tethering potential, and E_R the repulsive potential. In the following, we explain how we calculate these energies from triangulated surfaces.

To introduce the energy terms, we first have to define the DTS. A closed surface system consists of $N_T = 2(N_V - 2)$ triangles, with N_V vertices connected by $3(N_V - 2)$ tethers. To this end, we introduce a triangulation $\mathcal{T} := (\mathbf{x}, F)$ as a tuple of vertex positions $\mathbf{x} := \{x_i\}_{i=1}^{N_V}$ (with a single vertex $x_i \in \mathbb{R}^3$) and triangles $F := \{f_i\}_{i=1}^{N_T}$. Thereby a single oriented triangle $f_l := \{(i,j,k) | i,j,k \in [1, N_V], i \neq j \neq k\}$ is given by an ordered three-tuple indexing into the vertices $\mathbf{x}$. For the convenience of notation, we define the vertex–vertex connectivity $v_i^v := \{j | i,j \in f_k \forall f_k \in F, i \neq j\}$, the vertex–face connectivity $v_i^f := \{j | i \in f_j \forall f_j \in F\}$, the set of edges $E := \{(i,j) | i,j \in f_k \forall f_k \in F, i \neq j\}$ and the edge–face connectivity $e_{ij}^f := \{k | i,j \in f_k\}$.

We discretize the calculation of the Helfrich–Hamiltonian H_{tot} on the DTS using a vertex-averaged formulation as described in Ref. 19. However, other formulations exist and have been implemented for various simulation purposes,[17,31,35–37] each with its distinct benefits.

For $C_0 = 0$, the total bending energy E_B is given by the sum over all vertices of the bending energy per vertex

$$E_B = 2\kappa_B \sum_{i=1}^{N_V} \frac{\hat{M}_i^2}{\hat{A}_i}. \tag{3}$$

The area $\hat{A}_i$ per vertex i is calculated as

$$\hat{A}_i = \frac{1}{3} \sum_{j \in v_i^f} A_j, \tag{4}$$

where A_j is the area of the single triangle f_j. The average mean curvature $\hat{M}_i$ associated with vertex i can be calculated as sum over all adjacent edges

$$\hat{M}_i = \frac{1}{4}\sum_{j\in v_i^v} r_{ij}\phi_{ij}, \tag{5}$$

where $r_{ij} = \|x_j - x_i\|$ and ϕ_{ij} is the angle between the oriented normal vectors n_a, n_b of the triangles in e_{ij}^f, i.e., adjacent to the edge (i,j), calculated as $\cos(\phi_{ij}) = n_a \cdot n_b$.

The total mean curvature of the system is then computed as

$$M = \sum_{i=1}^{N_V} \hat{M}_i. \tag{6}$$

The energies for the volume and area constraint are given by E_V and E_A, respectively, and can be written as simple harmonic potentials

$$E_V = \kappa_V\left(\frac{V}{V_0} - 1\right)^2 \tag{7}$$

and

$$E_A = \kappa_A\left(\frac{A}{A_0} - 1\right)^2, \tag{8}$$

with κ_V, κ_A being the coupling constants, respectively. The corresponding reference values are given by V_0 and A_0. κ_V and κ_A are usually chosen several orders of magnitude larger than κ_B to restrict volume fluctuations to around 0.1% to model the near incompressibility of the lipids and the osmotic pressure differences, which lead to a near constant area-to-volume ratio.[20,24,25] The total area A is calculated as

$$A = \sum_{j=1}^{N_T} A_j. \tag{9}$$

Also, the volume V can be calculated as a discrete sum[19,25]

$$V = \sum_{i=1}^{N_T} V_i = \frac{1}{3}\sum_{i=1}^{N_T}(R_i \cdot n_i)A_i, \tag{10}$$

where V_i is the signed sub-volume of the tetrahedron formed by the origin and triangle i at position R_i with area A_i and outward-directed unit normal vector n_i.

The ADE, $E_{\Delta A}$, is given by

$$E_{\Delta A} = \kappa_{\Delta A}\left(\frac{\Delta A}{\Delta A_0} - 1\right)^2. \tag{11}$$

The area difference ΔA of a bilayer with respect to its shape can be written similarly as previously defined for continuous representations[38,39] as

$$\Delta A = 2h\sum_{i=1}^{N_V} \hat{M}_i, \tag{12}$$

where the sum runs over all vertices and $\hat{M}_i$ is the mean curvature associated with each vertex, respectively, and h is the thickness of the neutral surfaces. When choosing $\kappa_{\Delta A} \to \infty$, we recover the bilayer couple model where ΔA acts as a constraint.[39] Numerically, we convert this by choosing $\kappa_{\Delta A}$ to be on the order of 10^5–$10^6 k_B T$, where k_B is Boltzmann's constant and T is the temperature.[25]

The overall tether-energy is given by a coupling constant κ_T times the sum over the tether pair-potential $T(r_{ij})$ over all edge lengths r_{ij},

$$E_T = \kappa_T \sum_{(i,j)\in E} T(r_{ij}). \tag{13}$$

This energy serves to constrain the edge length r_{ij} and to guarantee the efficient and accurate simulation of fluid triangulated surfaces. Previous work[19,25] used discrete flat bottom potentials; however, these are not amenable to smooth time integration. In order to use the tether potential in hybrid MC simulations, a continuous representation is required.[20] To be able to use large integration time steps Δt, it is crucial to avoid diverging branches that are present in previous formulations.[20] Therefore, we use a continuous tether potential of the functional form

$$T(l) = \begin{cases} e^{\frac{l}{l-l_{c1}}} l^{-r} & \text{if} \quad l \le l_{c1}, \\ r^{r+1}(l - l_{c0})^r & \text{if} \quad l \ge l_{c0}, \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

with the l_{c1} and l_{c0} being the lower- and upper-onset of penalization and the slope $r \in \mathbb{N}_1$ of the potential well.

The overall repulsion energy is given by

$$E_R = \kappa_R \sum_{i=1}^{N_V} \sum_{j=1, j\notin S_i}^{N_V} R(r_{ij}), \tag{15}$$

where S_i is a set of excluded vertices for vertex i, e.g., the set of directly connected neighbors. The $\mathcal{O}(N_V^2)$ complexity involved in the evaluation of this potential is in practice reduced by the use of efficient neighbor tracking techniques.[40] Mimicking the repulsive electrostatic membrane–membrane interactions, mesh intersection is prevented by the introduction of a penalty that applies a repulsive force on pairs of non-bonded vertices that are below a certain threshold distance l_{c1}. Such a penalty is implemented by the repulsive branch of the tether penalty [Eq. (14)]. It is given by

$$R(l) = \begin{cases} e^{\frac{l}{l-l_{c1}}} l^{-r} & \text{if} \quad l \le l_{c1}, \\ 0 & \text{otherwise}. \end{cases} \tag{16}$$

The functional form of the membrane–membrane interaction could be expanded to model various properties of membrane adhesion and repulsion, and include the effects of proteins[41] or nanoparticles.[24]

III. ALGORITHMS AND IMPLEMENTATION FOR SIMULATION AND ENERGY MINIMIZATION

In the following, we discuss how to efficiently sample the configurational space of the DTS using the hybrid MC approach. First, we introduce the general sampling strategies and discuss how they are implemented. Next, we highlight the issues and solutions for efficient parallelization and evaluate their performance. Finally, we discuss important features to efficiently equilibrate and minimize structures in practice. The different algorithmic components introduced below are shown in the flowchart in Fig. 1. We apply these tools in Sec. IV.

A. General sampling strategy

The statistical properties associated with the Hamiltonian [Eq. (2)] follow the canonical distribution function given by

$$p(\mathbf{x}, F) \propto \exp[-\beta H_{\mathrm{tot}}(\mathbf{x}, F)], \tag{17}$$

where we use $\beta = 1/(k_B T)$ for the inverse temperature. We use a Markov chain MC procedure to sample configurations $(\mathbf{x}, F)$. To account for the compound nature of the state of the triangulation, given by vertex position as well as vertex connectivity, we generate new samples using the Metropolis algorithm with two randomly alternating, well established MC moves: (i) global vertex displacements and (ii) edge flips. After each step, the resulting configuration is accepted or rejected according to the Metropolis criterion[17] by evaluating the differences in the Hamiltonian H_{tot} as a function of $\mathbf{x}$ and F. Edge flips ensure membrane fluidity.

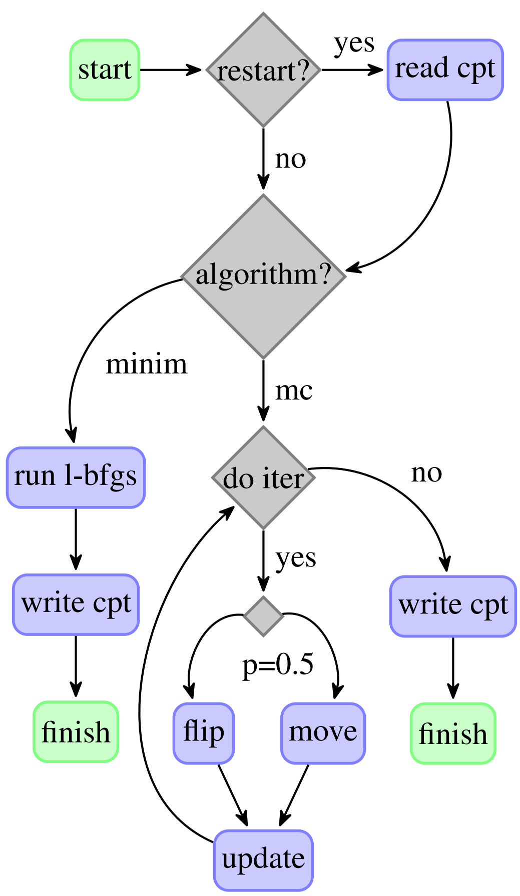

FIG. 1. Algorithmic flowchart of the TriMem software. "run l-bfgs" refers to Sec. III F, and "move" and "flip" are introduced in Algorithms 1 and 2, respectively. "update" summarizes the update of the reference parameters η (see Sec. III E) and the temperature T (see Sec. III G).

Vertex displacements are generated by a Hybrid Monte Carlo scheme (HMC).[32,42,43] HMC draws on the lifting of the vertex coordinates $\mathbf{x}$ onto an artificial phase-space $\mathbf{x} \mapsto (\mathbf{x}, \mathbf{p})$, where the components of the vector $\mathbf{p}$ are the momenta for all vertices. On this phase-space, we impose a symplectic structure via the lifted Hamiltonian

$$H(\mathbf{x}, \mathbf{p}) = H_{\mathrm{tot}}(\mathbf{x}) + \frac{1}{2}\mathbf{p}^{\top}\mathbf{M}^{-1}\mathbf{p}. \tag{18}$$

The transpose of a vector is indicated by superscript "$\top$" and $\mathbf{M}^{-1}$ is the inverse of the diagonal mass matrix $\mathbf{M}$. We exploit HMC to generate state transitions $(\mathbf{x}_n, \mathbf{p}_n) \to (\mathbf{x}_{n+1}, \mathbf{p}_{n+1})$ in the high-dimensional phase-space with high efficiency. Transitions generated in this way can then trivially be un-lifted $(\mathbf{x}_{n+1}, \mathbf{p}_{n+1}) \to \mathbf{x}_{n+1}$ to give a new sample $\mathbf{x}_{n+1}$. In practice, we use symplectic time-integration schemes, such as the leap-frog/Verlet integration method.[40] We show the general concept of a HMC step from $n \to n+1$ in Algorithm 1. Verlet-type integration accurately conserves $H(\mathbf{x}, \mathbf{p})$. As a result, the global vertex moves of HMC are accepted with probabilities close to one.

Conceptually, a HMC step consists of a short molecular dynamics (MD) simulation of the vertices, with velocities drawn from a Maxwell–Boltzmann distribution and forces given by the negative gradient of H_{tot} with respect to the vertex positions (see the Appendix). Subsequently, the lifted Hamiltonian is subject to the Metropolis criterion. The time integration parameter Δt and L can be used to tune the efficiency of the HMC algorithm. The time step Δt influences the acceptance probability within the Metropolis criterion. The number of steps L affects the mixing of the generated Markov chain. Tuning of these parameters is crucial for an efficient sampling scheme. The mass matrix $\mathbf{M}$ is another free parameter that can be tuned to optimize the sampling performance. In this work, we use a single mass m, $\mathbf{M} = m\mathbf{I}$, and set $m = 1$ as a default. This setting

ALGORITHM 1. One step of the HMC algorithm. We randomly draw new momenta from a Maxwell–Boltzmann distribution, perform a number of integration steps, and accept or reject the resulting configuration.

function HMC STEP $(x_n, \Delta t, L, T = 1)$
  $p_0 \sim \mathcal{N}(0, mT\mathbf{I})$
  $s_0 \leftarrow x_n$
  $i \leftarrow 0$
  while $i < L$ **do**
    $p_{i+1/2} = p_i - \frac{\Delta t}{2} \nabla H_{\mathrm{tot}}(x)|_{x=s_i}$
    $s_{i+1} = s_i + \Delta t M^{-1} p_{i+1/2}$
    $p_{i+1} = p_{i+1/2} - \frac{\Delta t}{2} \nabla H_{\mathrm{tot}}(x)|_{x=s_{i+1}}$
    $i \leftarrow i + 1$
  end while
  $\alpha \leftarrow \min(1, \exp(H(s_0, p_0) - H(s_L, p_L))/T)$
  $\sigma \sim \mathcal{U}_{[0,1]}$
  if $\sigma \le \alpha$ **then**
    return s_L
  else
    return x_n
  end if
end function

has proven efficient in practice. For sampling according to Eq. (17), the application of the HMC framework leads to an enormous gain in efficiency compared to a sequential single-vertex-move dynamic. In particular, in a high dimensional regime, i.e., large number of vertices N_V, HMC is essential for efficient sampling.

Vertex moves together with edge flip moves ensure the 2D-fluidity of the membrane and alleviate a possible bias due to a fixed vertex connectivity, i.e., vertices can change their neighborhood. In an edge flip, the shared edge (i,j) of two adjacent triangles (i,j,k) and (i,l,j) is changed to (k,l), resulting in modified triangles (i,l,k) and (j,k,l) (Fig. 2).

One step of edge flips consists of a sweep over a predefined fraction $\gamma \in [0,1]$ (in the following also referred to as target flip rate) of the set of edges E. It is conceptually depicted in Algorithm 2.

B. Implementation

The computational workload of the algorithms is dominated by the evaluation of the Hamiltonian and its gradient (see also the Appendix). This evaluation is needed from within the vertex moves as well as in the edge-flip move routine. At the core of the energy computation is the evaluation of the vertex-averaged quantities in Eqs. (4) and (5). Efficient evaluation of the vertex-connectivity is thus essential for overall performance. To this end, we use the concept of the half-edge data structure. This data structure for polygonal meshes efficiently tracks incidence information of edges, vertices and faces.[44,45] In particular, we make use of the OpenMesh library, which provides a generic implementation of the half-edge data structure in C++[46] that can handle various mesh sizes and geometries as input. To fully leverage the OpenMesh interface, we implement the evaluation of the Hamiltonian and its gradient in C++ and provide bindings to Python using the pybind11 Python package.[47] This approach combines the efficiency of a compiled programming language with the convenience of the well-established Python ecosystem for clear and extensible algorithm development. Using C++ for the computationally intensive evaluations further allows us to exploit shared-memory parallelism and the speedup offered by modern multi-core architectures. To achieve this, we use OpenMP[48] to parallelize the loops that involve scans or reductions over all vertices in the triangulation, such as Eqs. (6)–(10) and (12), and the respective gradient evaluations (see Sec. III D). While parallelization of both the vertex moves and components of the integration of short trajectories is straightforward, flip moves are more complex and require a more detailed discussion.

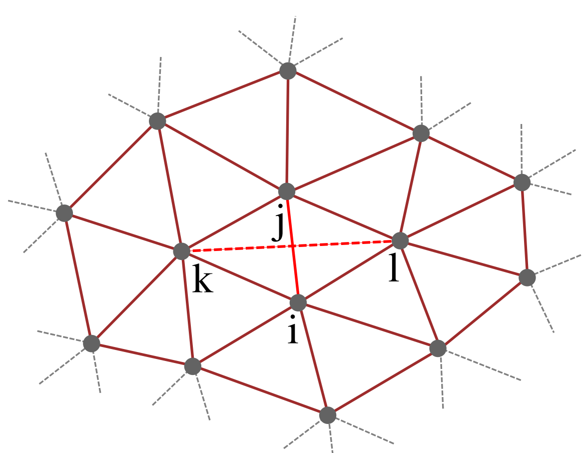

FIG. 2. Extent of an exemplary flip patch. Shown is a patch of vertices involved in a flip that is locked during a flip move. In a flip move, the edge (i,j) (light red, solid) is flipped to (k,l) (light red, dashed). All edges involved in the computation of the averaged vertex properties (dark red) for the vertices i, j, k, and l are included in the flip patch and must be locked during the flip operation. Note, for comparison, that this patch is significantly larger than the patch required for a flip subject to the Delaunay criterion[49] where the first shell neighbors are not required.

C. A strategy for the parallel evaluation of edge flips

To increase the computational efficiency of flip moves, we evaluate independent changes of geometric membrane properties and of the energy in parallel. We then evaluate sequentially the corresponding Monte Carlo flip moves, including the associated energy changes from the aforementioned pre-calculated quantities. We designed this Algorithm 3 to alleviate two of the main efficiency bottle necks of Algorithm 2: (i) flip execution, i.e., the technical realization of the change in vertex connectivity that is provided by the underlying data structure; and (ii) energy evaluation, i.e., the evaluation of the change in the Hamiltonian due to the change in connectivity. Both components are crucially influenced by the evaluation of the vertex connectivity.

Due to the edge-based connectivity information implemented by the half-edge data structure, flip execution provided by OpenMesh is realized efficiently by simply swapping edge connectivity.[46]

ALGORITHM 2. One sweep of edge flips. The input is the set of triangles F, the set of edges E, and the fraction γ of flips to attempt. It makes use of the Hamiltonian $H(F)$ as a function of the mesh triangles, and the routine flip_egde(e), which takes an edge $e \in E$, flips it, and alters the mesh connectivity. $\mathcal{U}_{\{n_l,n_o\}}$ and $\mathcal{U}_{[n_l,n_o]}$ represent discrete and continuous uniform distributions, respectively. The routine evaluate_edge_properties(e_i) evaluates the geometric properties M, A, and V and the independent contributions to the bending, tether, and repulsion energy. update_hamiltonian (h, F, p) incrementally updates the total energy using the pre-computed properties for the changed flip patch (see Fig. 2).

function FLIP SWEEP(F, E, γ)
    $h_n \leftarrow H(F)$
    for j in range $(\gamma|E|)$ **do**
        $i \sim \mathcal{U}_{\{0,|E|-1)\}}$
        flip_edge(e_i)
        $p_i \leftarrow$ evaluate_edge_properties(e_i)
        $h_{n+1} \leftarrow$ update_hamiltonian(h_n, F, p_i)
        $\alpha \leftarrow \min(1, \exp(h_n - h_{n+1}))$
        $\sigma \sim \mathcal{U}_{[0,1]}$
        if $\sigma \leq \alpha$ **then** (▷) evaluate Metropolis criterion
            $h_n \leftarrow h_{n+1}$
        else
            flip_edge(e_i) (▷) flip back in case of rejection
        end if
    end for
end function

Since the individual edge flip can technically be performed efficiently, the main computational workload results from the energy evaluation of the flip-patch associated with an edge flip. A straightforward exploitation of the speed-up provided by the evaluation of the change of the Hamiltonian for several edge flips in parallel is, however, still complicated by two intermingled issues.

First, individual edge flips tend to have very low acceptance probabilities due to the rather strict penalty on neighborhood distances that is imposed by the tether potential [Eq. (14)]. The evaluation of the Metropolis criterion for a flip of several edges at once thus suffers from an exponential decay of the acceptance probability due the multiplicative nature of the acceptance probabilities of single flips. Consequently, the acceptance of edge flips must be evaluated sequentially to establish a reasonable flip rate ε, which is defined as

$$\varepsilon = \frac{\text{number of accepted flips}}{\gamma|E|}, \tag{19}$$

where $|E|$ indicates the cardinality of the set of edges E. This sequential evaluation therefore represents an inherent serial component of Algorithm 2 that limits the potential speed-up by construction.

Second, selected edges have to be independent such that we can evaluate the respective changes in the Hamiltonian in parallel. Afterward, we sequentially evaluate the respective acceptance probabilities. The independence of edges means that they cannot be part of the same flip patch. A flip patch is the set of edges affected by a flip (see Fig. 2). This condition imposes a constraint on the set of edges for which the change in the Hamiltonian can be evaluated in parallel. In order to avoid the NP-hard problem of a deterministic pre-computation of sets of non-interfering edges, we propose a randomized approach that is outlined in the remainder of this section.

Inspired by the batch-parallel evaluation of mesh properties of Shang *et al.*,[50] we adopt a batch-parallel version of the flip-sweep (Algorithm 3). During the sweep-iteration, edges are selected at random by each thread from the entire set of edges E. To ensure independence of such a parallel batch of edges, each edge is attempted to be locked together with the associated flip-patch for the use by other threads. For technical reasons, we impose this strict condition of non-overlapping flip-patches within a batch of edges even though non-inclusion of a flipped edge in another patch would already suffice. The implementation draws upon a scoped data locking mechanism that allows for an efficient detection of patch clashes. Upon success of locking, a thread can independently evaluate the patch-local contributions to the geometric properties M [Eq. (6)], A [Eq. (9)], and V [Eq. (10)], as well as the contributions to the bending energy [Eq. (3)], the tether-penalty [Eq. (14)], and repulsion potential [Eq. (16)]. The computationally intense evaluations are thus parallelized over a batch of edges with batch size equal to the number of threads used in parallel. Subsequently, the change in the Hamiltonian and the acceptance probability for each edge in the batch are evaluated sequentially, thus preserving detailed balance of the resulting Markov chain at the level of batch iterations. Although this inherent serial component reduces the theoretically achievable speedup as mentioned above, it has shown to be satisfactory in practice while maintaining a flip rate ε close to the serial version (see Sec. III D for detailed results).

ALGORITHM 3. Parallel version of algorithm 2. The directives **# BARRIER** and **# CRITICAL** refer to OpenMP directives used in shared memory parallelism.[48] **# BARRIER** indicates a barrier where threads have to wait for each other. **# CRITICAL** defines a region that can only be executed by one thread at a time.

```
function FLIP SWEEP(F, E, γ)
    h_n ← H(F)
    # PARALLEL
    n_j ← γ|E|/ num_threads
    for k in range(n_j) do
        # BARRIER
        i ~ U_{0,|E|−1}
        locks ← lock_patch(e_i)
        # BARRIER
        if locks.empty() then
            Continue
        end if
        release_locks(locks)
        flip_edge(e_i)
        p_i ← evaluate_edge_properties(e_i)
        # CRITICAL {
        h_{n+1} ← update_hamiltonian(h_n, F, p_i)
        α ← min(1, exp(h_n − h_{n+1}))
        σ ~ U_[0,1]                  (▷) evaluate Metropolis criterion
        if σ ≤ α then
            h_n ← h_{n+1}
        else
            flip_edge(e_i)           (▷) flip back in case of rejection
        end if
        }                            (▷) end critical section
    end for
end function
```

D. Parallel performance

We analyze the parallel performance of the algorithms and methods introduced in Sec. III on a unit sphere with different degrees of mesh discretization as a test geometry. We use the icosahedron recursion technique from the trimesh Python package[51] to create high quality meshes with $N_V \in [624, 2562, 10\,242, 40\,962, 163\,842, 655\,362]$ vertices. The results referred to in this section are produced on a dual socket node with Intel® Xeon® Platinum 8280 CPUs with 56 cores in total.

Figure 3 shows results for the parallel scaling of the different algorithmic components with the number of threads for different numbers of vertices: energy evaluation [Eq. (2)], its gradient/force, flip-sweep according to Algorithm 3, and a full step of the MC-procedure outlined earlier, comprised of 1 HMC-step (Algorithm 1 with $L = 100$ and $\Delta t = 1.0 \times 10^{-4}$; note that the results in Figs. 3 and 4 are invariant with respect to the exact value of the time step Δt) plus 1 flip-sweep. To improve consistency, the measurements for energy, gradient, and flip-sweep comprise ten evaluations of the respective components due to their short run-time.

The high arithmetic intensity involved in the energy and gradient evaluations leads to efficient use of the available resources, which is reflected in good parallel speedup on the compute node. The flip-move sweep and MC-step evaluations also benefit from an increase

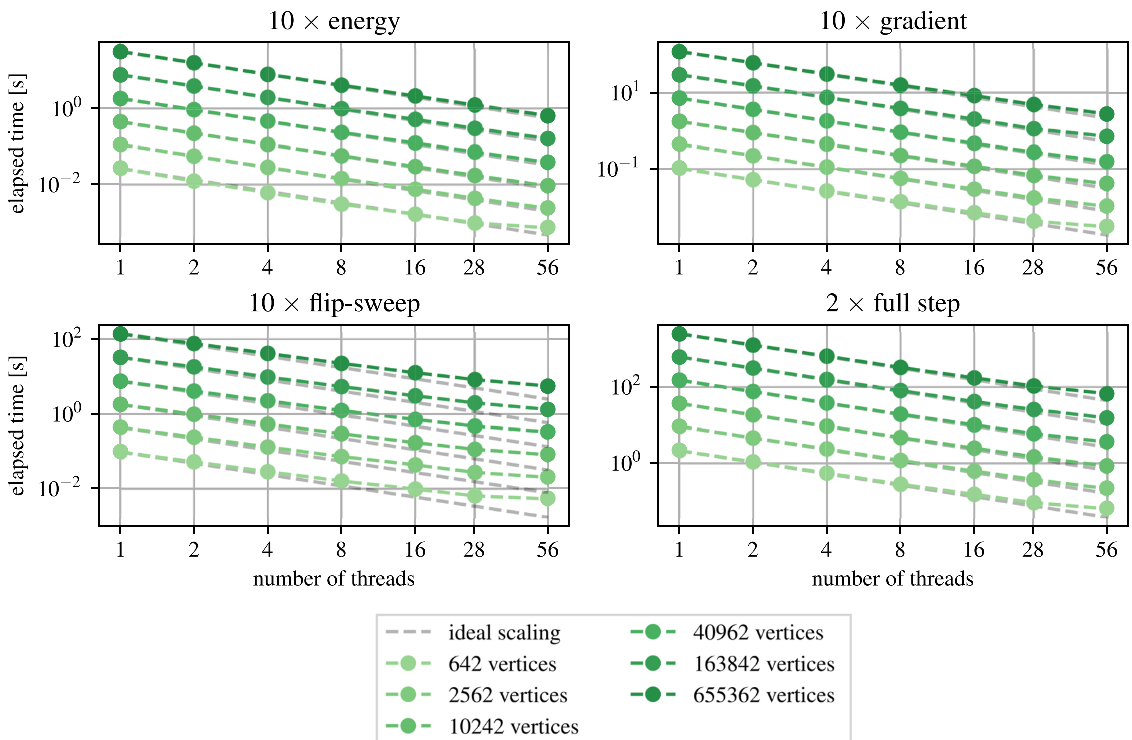

FIG. 3. Strong scaling of different algorithmic components with the number of threads for different problem sizes N_V (shades of green). The components are the evaluation of the energy, Eq. (2), its associated gradient/force, a flip-sweep according to Algorithm 3, and a full step of the MC-procedure outlined in Sec. III, however, with a deterministic succession of the HMC-step (with L = 100) and the flip sweep (γ = 0.1). For the components energy, gradient, and flip-sweep, measurements consist of ten evaluations each. The measurement for the component full-step consists of two steps.

in parallel execution performance, but a saturation trend is visible. This trend is a result of both of these components containing a significant amount of low arithmetic intensity workload. Thus, they are more exposed to the bound of the memory bandwidth than the energy and gradient evaluation. In addition, the flip-sweep still has a serial portion that will inherently limit the achievable speedup. However, in the tests presented here, this does not manifest itself as the critical component controlling the effective speedup. The scaling of the full MC-step is only minimally affected by the flip-sweep due to the small contribution of the flips-sweep to a full MC-step. Instead, it is governed by the amount of low arithmetic intensity workload in the HMC-step (Algorithm 1).

Importantly, with our parallelized scheme, the computation of a mesh with 40 962 vertices is comparable to a single-core run with a small system with 642 vertices. Without an explicit limit on mesh size in OpenMesh, it is possible, in principle, to simulate meshes at least up to a size of 10^6–10^7 vertices (Fig. 3), which would be required for cellular scale membranes. The global vertex moves in HMC ensures efficient sampling of the high dimensional space even for large mesh sizes.

The effective time complexity of the algorithmic components is shown in Fig. 4. The distance calculations necessary for the computation of the repulsion penalty given by Eq. (16) are effectively reduced to $\mathcal{O}(N_V)$ by the neighbor list data structures. This complexity on the level of the energy and gradient evaluation is directly passed on to the evaluation of the full MC-step.

The influence of the parallel implementation (Algorithm 3) on the flip rate ε of the flip-sweep is shown in Fig. 5. The measurements are carried out for a mesh with N_V = 10 242 vertices and a target flip rate γ = 10%. We found that the average flip rate of $\varepsilon \approx 0.17\%$ observed for the serial implementation reduces to ≈0.15% when using a full node with 56 cores. This slight decay is due to our randomized approach, in which the number of clashes in the locking of the necessary flip patches increases with the number of parallel threads used. In any case, the obtained flip rates have proven sufficient to provide the desired effect on the mesh mutability that is necessary to achieve membrane fluidity and accurate results, cf. Sec. IV.

E. Parameter continuation for stiff restraints on membrane shape

The accurate representation of the constraints regarding volume and surface area requires large values for the factors κ_V, κ_A and $\kappa_{\Delta A_0}$ in the respective penalty formulations given by Eqs. (7), (8), and (12). The conditioning of the Hamiltonian in Eq. (2) is determined by these penalty terms. That is, slight deviations of the initial

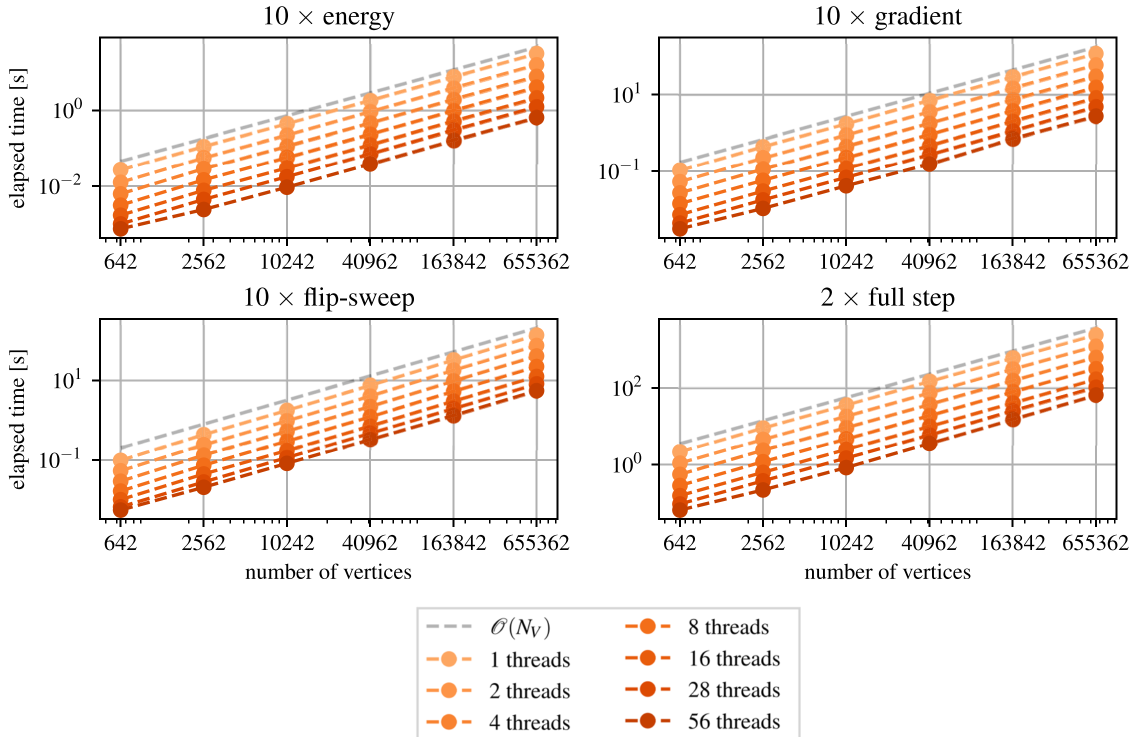

FIG. 4. Time complexity of different algorithmic components with the number of vertices N_V for different numbers of threads (shades of orange). The components are according to Fig. 3. For the components energy, gradient, and flip-sweep, measurements consist of ten evaluations each. The measurement for the component full-step consists of two steps. Color intensity indicates higher thread numbers. The slope of the gray dashed line indicates $\mathcal{O}(N_V)$ linear time.

configuration from the desired target configuration in terms of the area A and the volume V can have a large impact on the numerical stability of the Hamiltonian and the sampling performance. To mitigate such a performance degradation, TriMem offers

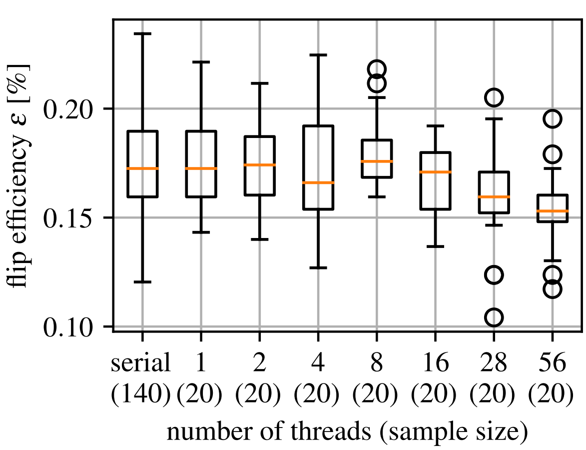

FIG. 5. Flip rate ε given by Eq. (19) of the parallel flip implementation given the number of threads. The target rate is $\gamma = 0.1$ and the number of vertices is $N_V = 10\,242$. For comparison, we show the rate for the serial version (leftmost data). The sample size is given in parentheses for every box. The flip efficiency decreases slightly with increased number of threads due to the locking of flip patches from different threads.

the possibility to use a technique from the concept of parameter continuation.[52] To this end, we introduce the parameterized Hamiltonian $H_p(\mathbf{x}, F; \eta)$, with the explicit definition of parameters $\eta := (V_0, A_0, \Delta A_0)$. This enables a smooth blending of the Hamiltonian $H_p(\mathbf{x}, F;\ \eta_0) \rightarrow H_p(\mathbf{x}, F;\ \eta_1)$ from some initial parameters η_0 to the target parameters η_1 via the linear interpolation

$$\eta = \lambda\eta_1 + (1 - \lambda)\eta_0 \tag{20}$$

by varying an interpolation parameter $\lambda \in [0, 1]$ smoothly from $0 \rightarrow 1$. By doing so, the sampling efficiency remains higher, even in situations in which the initial configuration is not consistent with the specific constraints in place. Parameter continuation can also help to overcome energy barriers that might appear in an immediate instantiation of the target Hamiltonian. More generally, the parameterized Hamiltonian allows us to integrate systematic approaches to branch tracking[52] in future work.

Currently, we interpolate the whole set of parameters η simultaneously by defining and implementing a single interpolation parameter λ. The efficiency of this scheme could be improved by using a vector of interpolation parameters such that individual components of the Hamiltonian are transformed independently.

F. Gradient-based energy minimization

If the initial configuration is not consistent with the imposed constraints, we can improve the sampling efficiency by initial energy minimization. Such a minimization can be interpreted as the one-time application of a preconditioning and will bring the initial configuration closer to the equilibrium configuration, determined by Eq. (17). Energy minimization also avoids the necessity of running long simulations with parameter interpolation in the beginning. Since this is only an initial energy minimization prior to a HMC simulation, no particular requirements must be imposed on the convergence to the global optimum. By ignoring edge flips during minimization, we can use well established and efficient methods for function optimization such as the L-BFGS method.[53] In TriMem, simulations can be run in minimization or HMC mode (see Fig. 1).

G. Simulated annealing

Gradient-based energy minimization of the vertex positions must be followed by a global minimization strategy accounting for both vertex positions as well as mesh connectivity. We apply a simulated annealing procedure for the exploration of the domain of Eq. (17) that is capable of finding global minima/maxima. Following Ref. 54, we implement this method by simply modifying the temperature argument of the HMC step in Algorithm 1 according to a cooling schedule. Specifically, we apply an exponential cooling scheme $T_{n+1} = \max[\exp(-\lambda)T_n, T_{\min}]$, with the cooling factor $\lambda \in \mathbb{R}^+$ controlling the rate of cooling.

In TriMem, cooling can be initiated during HMC simulations directly in the beginning or after a longer equilibration period prior to the cooling. This enables sufficient sampling of the configurational space before settling into the global (or deep local) energy minimum.

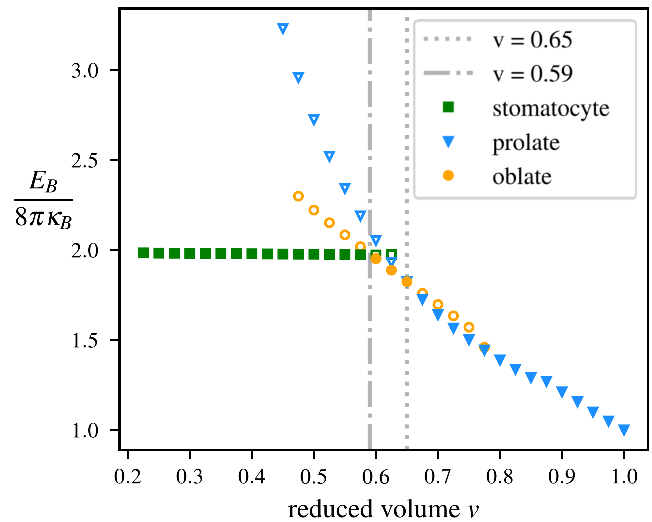

FIG. 6. Phase diagram for a vesicle with $N_v = 1962$ vertices of spherical topology in the plane of the reduced volume v and the bending energy E_B. Shown are minimum energies obtained for different membrane shapes as a function of v. Filled symbols indicate the respective lowest-energy shapes. Open symbols indicate metastable shapes at a given reduced volume. All states with prolate shapes are shown as blue triangles. Oblate shapes are shown as orange circles. Stomatocytes are shown as green rectangles. The dotted-dashed and dashed lines correspond to $v = 0.59$ and 0.56, respectively, and mark transitions between stable branches. Exemplary energy-minimized shapes for all corresponding branches are shown in Fig. S1.

IV. VALIDATION METHODS

We tested the robustness of the TriMem software and compared it to analytical and numerical calculations from the literature. We reproduced several well established aspects of the phase diagram for closed vesicles with $C_0 = 0$ and with respect to varying volume or area difference. In all simulations, we used a bending rigidity of $\kappa_B = 30k_BT$, which is a typical value for biological membranes. The coupling constants of the volume, κ_V, the area, κ_A, and the area difference, $\kappa_{\Delta A}$, were chosen several orders of magnitude larger with $\kappa_V = \kappa_A = \kappa_{\Delta A} = 1 \times 10^6 k_BT$ to impose a strong restraint. All simulations were performed using a mesh size of $N_V = 1962$ or 642 vertices starting from a sphere. The initial shapes were generated with the trimesh Python library.[51] Our results are not affected by mesh size.

A practical instruction to set up and run such simulations with TriMem is available via Github (https://github.com/biophys/trimem) on the documentation webpage.

A. Volume phase diagram

The individual configurations of the branches of the volume phase diagram (Fig. 6) were generated as follows: The initial shapes for each branch were generated using the minimization procedure to initialize the system. Prolate simulations were started at the reduced volume $v = 3V/(4\pi R^3) = 1.0$ (Fig. 6; blue triangles), oblate simulations at volume $v = 0.65$ (orange circles) and stomatocyte simulations at $v = 0.3$ (green squares). These initial shapes and volumes on the respective branches were achieved by using the preconditioning procedure (see Sec. III F). Then, using these initial shapes, the reference reduced volume v was lowered or increased by 0.025 instantaneously in each step. In each step, the previous final structure was used as initial configuration for the subsequent step. By doing so, the respective branches could be mapped out by exploiting hysteresis. In all cases the simulations could, in principle, switch their respective branches on the phase diagram, and otherwise, no special restraint was applied. Each simulation was run for 5×10^5 steps using a temperature $T = 1k_bT$ and in an additional 6×10^5 steps the temperature was reduced from 1 to 0 following Sec. III G. The rescaled bending energy of the lowest energy configuration was plotted for each value. In all HMC simulations, an integration step size of 7^{-5} was used. The trajectories in each HMC step were 100 steps long. The flip ratio was set to $\gamma = 0.1$. Previous tests showed this flipping ratio was sufficient for fast vertex diffusion.

B. Area-difference phase diagram

The area-difference phase diagram is another commonly used benchmark to test the robustness of numerical membrane bending simulation codes.[25,31,55] The simulations were performed as follows: The initial shape for the HMC run was generated in a two-step minimization procedure. First, the volume was reduced to the respective target reduced volumes of $v = \{0.5, 0.55, 0.6\}$, and second, the reduced area difference $\Delta a = \Delta A/(4\pi R)$ was adjusted to the respective target values between 1 and 1.8, by using the L-BFGS minimizer for gradient based energy minimization for both parameters successively. Then, a HMC simulation was run for 1.5×10^7 steps. After 1.4×10^7 steps 1×10^6 steps of cooling were performed where the temperature was reduced from 1 to 0 according to Sec. III G. The bending energy and shape at the final step of each simulation were extracted and normalized by $8\pi\kappa_B$ to obtain the reduced energy and construct the phase diagram. Otherwise, the same parameters were used for the HMC simulations as described in Sec. IV A.

C. Test for Boltzmann sampling

To calculate the density of states and verify Boltzmann sampling, HMC simulations for a vesicle with reduced volume $v = 0.95$ and 642 vertices were performed at nine different temperatures of $T = 1, 1.25, 1.5, \ldots, 3$. The same parameters as described in Sec. IV A were used for the HMC simulations. Each HMC simulation was run for 5×10^6 steps, and the total energy H_{tot} of the configurations was written out every 50 steps. The first 10^6 steps of the simulation were not included in the analysis. The resulting energies of each simulation were binned in unit intervals. From these histograms, the density of states was estimated. In a Boltzmann distribution at inverse temperature $\beta = 1/(k_{\mathrm{B}}T)$, the potential energies H_{tot} are expected to be distributed as $p(H_{\mathrm{tot}}|\beta) = \Omega(H_{\mathrm{tot}})e^{\beta(G_\beta - H_{\mathrm{tot}})}$ with $\Omega(H_{\mathrm{tot}})$ the density of states (DOS) defined up to a factor and independent of β, and G_β the β-dependent Helmholtz free energy defined up to a constant. Therefore, in a plot of $\ln p(U|\beta) + \beta(H_{\mathrm{tot}} - G_\beta)$ as a function of H_{tot}, the curves from different temperatures are expected to coincide, accounting for some scatter at the respective tails. The free energies G_β defining the vertical offsets of the different curves were found by weighted histogram analysis.[56]

V. RESULTS AND DISCUSSION

We extensively validated our software for varying mesh sizes and exhaustively tested all features by reproducing known phase diagrams of vesicle shape.[55,57] First, we ran a range of simulations to produce the volume phase diagram for a vesicle shown in Fig. 6. In the following, we consistently use reduced volumes v as an order parameter. Starting with a sphere ($v = 1$), we explored the range between $v = 1$ and 0.2 to locate minimal shapes found as described in Sec. IV. The y-axis shows the bending energy E_B and was rescaled with $8\pi\kappa_B$, the bending energy of a sphere, as in previous work.[25,55]

Here, we explored the stability of the three well established branches in the vesicle volume phase diagram; the prolate, the oblate, and the stomatocyte.[19,31] The transitions between the three branches are not smooth, i.e., the first derivative of the energies with respect to the reduced volume are discontinuous. Stable and metastable states representing the various shapes are separated by energy barriers.

The prolate branch (Fig. 6, blue triangles) gives the global energy minima for $v \gtrsim 0.65$. As the reduced volume is lowered, the vesicle changes its shape from the sphere at $v = 1$ to a prolate shapes and then to extended tubular structures. Low reduced volumes result in long metastable tubes with narrow diameter and a far higher bending energy than the corresponding oblate or stomatocyte shapes with the same reduced volume, which constitutes the global minimum for $v \lesssim 0.65$. Below this reduced volume the branch is metastable as previously outlined in Refs. 25 and 31. All prolate shapes and tubes below $v = 0.65$ are only found when starting from an initial prolate configuration and mapping out the branch by hysteresis, indicating that they are local minima and metastable.

The oblate branch is the global minimum between $0.59 \lesssim v \lesssim 0.65$ (Fig. 6, orange circles). Simulations starting from the sphere instantaneously reducing the volume and using the L-BFGS minimizer to a target value $v < 0.65$ always converged to an oblate shape. This result is in agreement with observations of previous work.[19,31] Above $v = 0.725$, the oblate shape is always unstable and the local energy minimum for the branch vanishes, and all initial shapes (sphere, prolate, and oblate) converged to the prolate shape, which is consistent with previous theoretical work.[58] Oblates in the range of $0.65 < v \lesssim 0.725$ can only be seen when starting with an oblate configuration in the globally stable range and tracing out the branch by hysteresis. By doing so the structures remaining in the metastable minimum. Similarly, the oblate shape is metastable at a reduced volume of $v \lesssim 0.59$.

For a reduced volume of $v \lesssim 0.59$ the stomatocyte is the global minimum, which is also achieved from the preconditioning step with the L-BFGS minimizer from a sphere using a stiff restraint on the target volume (Fig. 6 green squares). For $v > 0.59$, we find that the stomatocyte shapes are metastable with respect to oblates. The energy of the stomatocytes are nearly independent of the reduced volume with $E_B \approx 16\pi\kappa_B$. Overall, all energies for the globally energy-minimized shapes are in good agreement with previous work following similar approaches.[19,31]

To test if the area-difference is correctly evaluated, we also produced the phase diagram for varying area differences at multiple fixed reduced reduced volumes. Figure 7 shows the rescaled bending energy as a function of the area difference Δa_0. Simulations were run for area-difference increments of 0.025 and the energy-minimized structures after cooling to $T = 0$ were plotted (see Sec. IV).

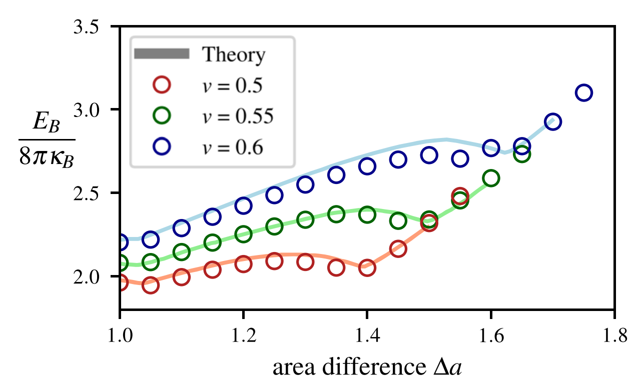

FIG. 7. Energy as function of reduced area difference Δa at constant volume v for a mesh with $N_v = 642$. The simulation results are shown as empty circles for a constant volume $v = 0.5$ (blue), 0.55 (green), and 0.6 (red), respectively. For reference, the light lines are adapted from earlier calculations[55] using numerical triangulation.[59] Exemplary energy-minimized shapes for varying area difference are shown in Fig. S2.

We verified that both the minimized energies and the corresponding shapes are in agreement with previous simulations and theoretical calculations.[25,31,55] Small differences in the bending energies for some shapes in Fig. 7 are likely caused by hysteresis effects around boundaries in shape space, the use of edge flips here, and different resolutions of the triangulations. The oblate configuration and the tube shapes are metastable for $v = 0.55$ and are separated by an energy barrier. The top of the barrier corresponds to the non-axisymmetric paddle shape that separates the prolate and tube branch and converts to a tube with increasing Δa. For all values with $\Delta a < 1.4$, we observed structures with a D_{3h} symmetry. First triangular shaped oblates emerge, which transition into three-armed starfish shapes (Fig. S2). We also find that the dumbbell/tube structure[55] is a local minimum for $\Delta a_0 > 1.4$. Tubes are found for all higher values of Δa_0. We note that extensive descriptions and discussions of these phase diagrams and their symmetries are provided in previous systematic studies.[25,31,55]

We tested the code for correct Boltzmann sampling by performing simulations at a reduced volume $v = 0.95$ over a wide range of temperatures $(1 \leq T \leq 3)$. We then generated the histograms of $p(H_{\mathrm{tot}}|\beta)$ [Fig. 8(a)] that had ample overlap in the distributions. These simulations indeed produce a fully consistent density of states $\Omega(H_{\mathrm{tot}})$ independent of temperature [Fig. 8(b)]. After the transformation $\ln p(H_{\mathrm{tot}}|\beta) + \beta(H_{\mathrm{tot}} - G_\beta)$, with G_β the Helmholtz free energy obtained by weighted histogram analysis,[56] these distributions are fully coincident far into the tails, as is expected for Boltzmann sampling. Figure 8 shows that the coincidence extends over multiple histograms. For instance, the yellow points $(T = 2.5)$ for $H_{\mathrm{tot}} \approx 1600$ and 1800, respectively, whose tails are visible in the two vertically displaced curves, superimpose with the transformed histograms from $T = 1.75$ to 3.0. Given this excellent consistency obtained for independent simulations over a broad range of temperatures, this analysis provides a powerful test for Boltzmann sampling.

Taken together, these simulations validate that TriMem faithfully reproduces data from previous studies and can be used to study a broad range of reshaping processes. All parameters are correctly assessed and our parallel dynamics protocol produces the expected results. It can be used both for minimization problems as well as thermodynamic sampling problems, as demonstrated in our test cases.

VI. CONCLUSIONS AND OUTLOOK

We presented TriMem, a parallelized open-source software package for HMC simulations of triangulated meshes representing lipid membranes. TriMem offers a robust, efficient, and parallel implementation of the method that enables users to perform simulations efficiently with large mesh sizes of 10^6–10^7 vertices, which was not previously possible. Many previously available software packages were limited to a few thousand vertices by the computational cost. We demonstrated how all relevant move types can be efficiently parallelized across multiple CPU cores up to a full two-socket shared memory compute-node (Fig. 3). While parallelization of energy and force routines to generate short MD trajectories for vertex moves is straightforward, we showed that flip parallelization requires a more complex algorithm to produce correct results. We introduced a novel method for flip moves that combines an effective generation of sets of edges with non-interfering flip-patches. The parallel computation of the independent energy changes and subsequent evaluation of the acceptance in series increases the performance. We bypass a major issue resulting from serial energy evaluation required for individual flip moves.

Additionally, the HMC approach allows for more efficient global vertex displacement moves with acceptance ratios of nearly one due to efficient proposals generated by short molecular dynamics trajectories of the mesh and vertices.[20] This leads to fast convergence of sampling and will be particularly useful when simulating large meshes with millions of vertices.

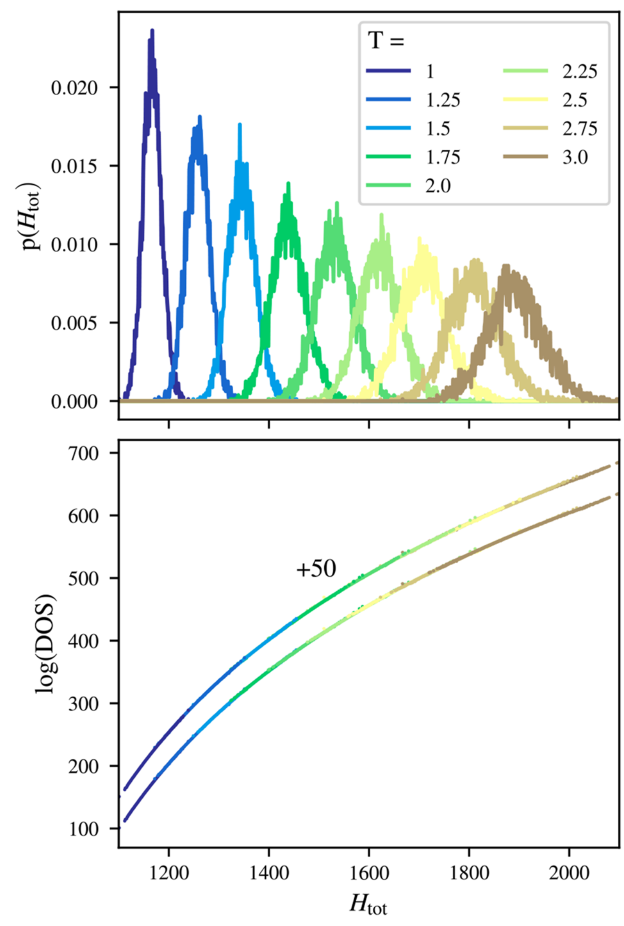

FIG. 8. Density of states is consistent with Boltzmann sampling. (Top) Histograms $p(H_{\mathrm{tot}}|\beta)$ of the potential energy H_{tot} obtained in simulations of a vesicle with reduced volume $v = 0.95$ and 642 vertices at temperatures $1 \leq T \leq 3$ (left to right). (Bottom) Logarithm of the density of states. The transformation $\ln p(H_{\mathrm{tot}}|\beta) + \beta(H_{\mathrm{tot}} - G_\beta)$, with G_β the Helmholtz free energy, brings the nine histograms in the top plot into coincidence. The resulting curve defines, up to a constant, the logarithm of the density of states. The top curve is shifted vertically by an arbitrary constant of 50 and has the histograms at higher temperature plotted first (and thus "below") to highlight the right tail of the distributions.

We implemented an efficient neighbor-list algorithm to enable the use of repulsive, and possibly adhesive potentials, between non-neighbor vertices. The resulting code scales linearly, $\mathcal{O}(N_V)$, with mesh size, and makes calculations with large meshes possible. This feature proved particularly important to sample highly curved membrane shapes such as stomatocytes, where self-intersection of the mesh could result in faulty representations. It can also be further expanded to model the general physical properties of membrane repulsion due to charges, possible protein-mediated adhesion, and systems with multiple membrane meshes.

The user interface of the software was designed for versatility but with ease-of-use in mind. Some functionality, such as the configuration file, is kept loosely in the style of well-established molecular simulation packages, such as GROMACS,[60] which is widely used in the biophysics community. The ease-of-use and open-source nature of the code should empower a broad community of biophysicists to quickly set up, run, and evaluate large-scale membrane remodeling processes of interest. The Python front-end should also enable other users to quickly add novel features to the code.

We validated the software by reproducing several phase diagrams and results for vesicle shaping, which have been previously explored in various software packages and foot on theoretical calculations.[19,25,31,55,58] We found in all cases that the validation was in good agreement with existing work, indicating the robustness of our algorithms.

The code was designed for performance with C++ in the backend and for ease-of-use with a Python interface in the frontend. The framework provided by the TriMem package can be extended in the future in a straightforward way to enable a broader range of large-scale membrane simulations and possible applications to complex biological systems. Various extensions for the Helfrich model have been previously reported and can be consolidated, parallelized, and made easily available through TriMem.[7,27–29,36] Systems containing multiple membranes in close vicinity are needed to build organelles and cell-scale membrane structures.[27] Additionally, protein vertices and lipid domains can be included to study complex protein-induced membrane remodeling processes.[7,28,36]

We anticipate that TriMem, with some modifications, will also enable simulations of complex membrane topologies. One possible direction is to include periodic boundary conditions (PBC) when defining the topology through the local connectivity of the graph. An implementation of PBC will enable the preparation of quasi-infinite membrane shapes from tubes (PBC in z direction) over flat membranes as previously reported[29] (PBC in the x–y plane) to lipidic mesophases (PBC in all directions x, y, z).

Another possible direction is to simulate multiple disconnected membranes, such as one vesicle enclosing another, or stacks of membranes as they occur in the Golgi.[27] Here, the vertex repulsion can ensure that the different membranes do not interpenetrate. It is also possible to fix the spatial position of certain vertices. In this way, the dynamic membrane can be connected at fixed seams to other shapes, e.g., to form the half-toroidal pores connected with two flat background membranes as seen in nuclear pore complexes.[13] Finally, it might also be possible to study efficiently some topology changes where Gaussian curvature is relevant.[61]

In the future, we envision that TriMem in combination with modern structural biology techniques such as cryo-ET and further computational advancements will shed light on the structure and dynamics of membranes on an organelle and cell-scale level. Cryo-ET will increasingly serve as data source to build accurate computational models for coarse-grained simulations. We envision a combination of multi-scale simulation techniques including TriMem's parallelized capability will be helpful for performing large-scale simulations required to gain mechanistic understanding of complex cellular membrane remodeling processes.

VII. SOFTWARE AVAILABILITY

TriMem is available as free software under a GPL-3 license from (https://github.com/bio-phys/trimem) and can be installed as a Python package using *pip*.[62] As such, it can be used as a Python library with predefined building blocks for the methods and algorithms presented in the previous sections. It also provides a command line interface that encapsulates the algorithmic flow shown in Fig. 1, which is controlled via an input configuration file. For a more detailed description on the usage, we refer the reader to the documentation which can be found via the Github link above. Software contributions from third party developers are welcome as pull requests on Github.

SUPPLEMENTARY MATERIAL

See the supplementary material for Figs. S1 and S2 showing minimum-energy vesicle shapes.

ACKNOWLEDGMENTS

This work was supported by the Max Planck Society. G.H. and M.S. acknowledge funding by the Landes-Offensive zur Entwicklung Wissenschaftlich-Ökonomischer Exzellenz LOEWE of the State of Hesse (https://wissenschaft.hessen.de/wissenschaft/landesprogramm-loewe).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

M.S. and S.K. contributed equally.

Marc Siggel: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Software (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Sebastian Kehl**: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Project administration (equal); Software (equal); Validation (equal); Visualization (equal); Writing –

original draft (equal); Writing – review & editing (equal). **Klaus Reuter**: Methodology (equal); Software (equal); Writing – review & editing (equal). **Jürgen Köfinger**: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Project administration (equal); Supervision (equal); Writing – review & editing (equal). **Gerhard Hummer**: Conceptualization (equal); Funding acquisition (equal); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX: GRADIENT OF THE HAMILTONIAN

The gradient of the Hamiltonian Eq. (2) can be derived in terms of the derivatives of quantities defined on the n-simplexes that define the triangulation $\mathcal{T}$. For the derivatives of these quantities with respect to the vertex positions x_i, we apply efficient and well-known formulas from discrete geometry.[63] In particular, we need the derivatives of the edge length, the face area, the face volume and the dihedral angle with respect to the involved vertex coordinates.

For the length $r_{ij} = \|u\| = \|x_j - x_i\|$ of the directed edge (x_i, x_j), the gradient is given by

$$\frac{\mathrm{d}r_{ij}}{\mathrm{d}x_i} = -\frac{u}{\|u\|}, \tag{A1}$$

$$\frac{\mathrm{d}r_{ij}}{\mathrm{d}x_j} = \frac{u}{\|u\|}. \tag{A2}$$

The gradient of the area A of an oriented triangle $(i, j, k) \in F$ with normal n is given by

$$\frac{\mathrm{d}A}{\mathrm{d}x_i} = \frac{1}{2} n \times (x_k - x_j), \tag{A3}$$

$$\frac{\mathrm{d}A}{\mathrm{d}x_j} = \frac{1}{2} n \times (x_i - x_k), \tag{A4}$$

$$\frac{\mathrm{d}A}{\mathrm{d}x_k} = \frac{1}{2} n \times (x_j - x_i). \tag{A5}$$

The gradient of the volume V associated with an oriented triangle $(i, j, k) \in F$ (computed as the volume of the tetrahedron $(x_i, x_j, x_k, \mathcal{O})$, $\mathcal{O}$ being the origin, is given by

$$\frac{\mathrm{d}V}{\mathrm{d}x_i} = \frac{1}{6}(x_j \times x_k), \tag{A6}$$

$$\frac{\mathrm{d}V}{\mathrm{d}x_j} = \frac{1}{6}(x_k \times x_i), \tag{A7}$$

$$\frac{\mathrm{d}V}{\mathrm{d}x_k} = \frac{1}{6}(x_i \times x_j). \tag{A8}$$

And the gradient of the dihedral angle ϕ_{ij} between the normals of the two oriented triangles $(i, j, k) \in F$ and $(i, j, l) \in F$, with common edge (i, j) and normals n_1 and n_2, is given by

$$\frac{\mathrm{d}\phi_{ij}}{\mathrm{d}x_i} = (\cot(\alpha_3)n_1 + \cot(\alpha_4)n_2)/r_{ij}, \tag{A9}$$

$$\frac{\mathrm{d}\phi_{ij}}{\mathrm{d}x_j} = (\cot(\alpha_1)n_1 + \cot(\alpha_2)n_2)/r_{ij}, \tag{A10}$$

$$\frac{\mathrm{d}\phi_{ij}}{\mathrm{d}x_k} = -(\cot(\alpha_1) + \cot(\alpha_3))n_1/r_{ij}, \tag{A11}$$

$$\frac{\mathrm{d}\phi_{ij}}{\mathrm{d}x_l} = -(\cot(\alpha_2) + \cot(\alpha_4))n_2/r_{ij}, \tag{A12}$$

where the α_i are the sector angles defined by

$$\alpha_1 := \angle x_k x_i x_j, \tag{A13}$$

$$\alpha_2 := \angle x_j x_i x_l, \tag{A14}$$

$$\alpha_3 := \angle x_i x_j x_k, \tag{A15}$$

$$\alpha_4 := \angle x_l x_j x_i. \tag{A16}$$

REFERENCES

1. H. T. McMahon and J. L. Gallop, Nature **438**, 590 (2005).
2. H. T. McMahon and E. Boucrot, J. Cell Sci. **128**, 1065 (2015).
3. J. Mahamid, S. Pfeffer, M. Schaffer, E. Villa, R. Danev, L. Kuhn Cuellar, F. Förster, A. A. Hyman, J. M. Plitzko, and W. Baumeister, Science **351**, 969 (2016).
4. S. J. Marrink, V. Corradi, P. C. T. Souza, H. I. Ingólfsson, D. P. Tieleman, and M. S. P. Sansom, Chem. Rev. **119**, 6184 (2019).
5. 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, and S. J. Marrink, J. Am. Chem. Soc. **136**, 14554 (2014).
6. M. Chavent, A. L. Duncan, and M. S. Sansom, Curr. Opin. Struct. Biol. **40**, 8 (2016).
7. N. Ramakrishnan, P. B. Sunil Kumar, and R. Radhakrishnan, Phys. Rep. **543**, 1 (2014).
8. W. Pezeshkian, M. König, S. J. Marrink, and J. H. Ipsen, Front. Mol. Biosci. **6**, 1 (2019).
9. W. Pezeshkian and S. J. Marrink, Curr. Opin. Cell Biol. **71**, 103 (2021).
10. I. R. Cooke, K. Kremer, and M. Deserno, Phys. Rev. E **72**, 011506 (2005).
11. B. J. Reynwar, G. Illya, V. A. Harmandaris, M. M. Müller, K. Kremer, and M. Deserno, Nature **447**, 461 (2007).
12. M. Siggel, R. M. Bhaskara, M. K. Moesser, I. Ðikić, and G. Hummer, J. Phys. Chem. Lett. **12**, 1926 (2021).
13. S. Mosalaganti, A. Obarska-Kosinska, M. Siggel, R. Taniguchi, B. Turoňová, C. E. Zimmerli, K. Buczak, F. H. Schmidt, E. Margiotta, M.-T. Mackmull, W. J. H. Hagen, G. Hummer, J. Kosinski, and M. Beck, Science **376**, eabm9506 (2022).
14. M. Vendruscolo and C. M. Dobson, Curr. Biol. **21**, R68 (2011).
15. W. Helfrich, Z. Naturforsch., C **28**, 693 (1973).
16. W. Helfrich, Z. Naturforsch., C **29**, 510 (1974).
17. G. Gompper and D. M. Kroll, J. Phys.: Condens. Matter **9**, 8795 (1997).
18. J.-S. Ho and A. Baumgärtner, Phys. Rev. Lett. **63**, 1324 (1989).
19. F. Jülicher, J. Phys. II **6**, 1797 (1996).
20. H. Noguchi and G. Gompper, Phys. Rev. E **72**, 011901 (2005).
21. B. Różycki, E. Boura, J. H. Hurley, and G. Hummer, PLoS Comput. Biol. **8**, e1002736 (2012).
22. F. Jülicher and R. Lipowsky, Phys. Rev. Lett. **70**, 2964 (1993).
23. A. Šarić and A. Cacciuto, Phys. Rev. Lett. **108**, 118101 (2012).
24. A. H. Bahrami, R. Lipowsky, and T. R. Weikl, Phys. Rev. Lett. **109**, 188102 (2012).
25. A. H. Bahrami and G. Hummer, ACS Nano **11**, 9558 (2017).
26. A. H. Bahrami, M. G. Lin, X. Ren, J. H. Hurley, and G. Hummer, PLoS Comput. Biol. **13**, e1005817 (2017).
27. M. Tachikawa and A. Mochizuki, Proc. Natl. Acad. Sci. U. S. A. **114**, 5177 (2017).
28. W. Pezeshkian, A. G. Hansen, L. Johannes, H. Khandelia, J. C. Shillcock, P. B. S. Kumar, and J. H. Ipsen, Soft Matter **12**, 5164 (2016).

[29] W. Pezeshkian and J. H. Ipsen, Soft Matter **15**, 9974 (2019).
[30] C. Zhu, C. T. Lee, and P. Rangamani, Biophys. Rep. **2**, 100062 (2022).
[31] X. Bian, S. Litvinov, and P. Koumoutsakos, Comput. Methods Appl. Mech. Eng. **359**, 112758 (2020).
[32] M. E. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation* (Oxford University Press, Oxford, 2010).
[33] U. Seifert, Adv. Phys. **46**, 13 (1997).
[34] K. Berndl, J. Käs, R. Lipowsky, E. Sackmann, U. Seifert, J. Kas, R. Lipowsky, E. Sackmann, and U. Seifert, Europhys. Lett. **13**, 659 (1990).
[35] N. Ramakrishnan, P. B. Sunil Kumar, and J. H. Ipsen, Phys. Rev. E **81**, 041922 (2010);arXiv:1004.4509.
[36] N. Ramakrishnan, P. B. Sunil Kumar, and J. H. Ipsen, Biophys. J. **104**, 1018 (2013).
[37] W. Pezeshkian and J. H. Ipsen, Phys. Rev. E **103**, L041001 (2021).
[38] S. Svetina and B. Žekš, Eur. Biophys. J. **17**, 101 (1989).
[39] L. Miao, U. Seifert, M. Wortis, and H.-G. Döbereiner, Phys. Rev. E **49**, 5389 (1994).
[40] L. Verlet, Phys. Rev. **159**, 98 (1967).
[41] Y. C. Kim and G. Hummer, J. Mol. Biol. **375**, 1416 (2008).
[42] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, Phys. Lett. B **195**, 216 (1987).
[43] B. Mehlig, D. W. Heermann, and B. M. Forrest, Phys. Rev. B **45**, 679 (1992).
[44] M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf, "Polygon triangulation," *Computational Geometry: Algorithms and Applications* (Springer, Berlin, Heidelberg, 2000), pp. 45–61.
[45] L. Kettner, Comput. Geom. **13**, 65 (1999).
[46] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, in OpenSG Symposium, 2002.
[47] W. Jakob, J. Rhinelander, and D. Moldovan, "pybind11—Seamless operability between C++11 and Python," 2017, https://github.com/pybind/pybind11.
[48] OpenMP Architecture Review Board, "OpenMP application program interface version 5.2," 2021, https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf.
[49] B. Delaunay, Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles **6**, 793–800 (1934).
[50] M. Shang, C. Zhu, J. Chen, Z. Xiao, and Y. Zheng, Procedia Eng. **163**, 289 (2016) 25th International Meshing Roundtable.
[51] M. Dawson-Haggerty *et al.*, "trimesh," 2019, https://trimsh.org/.
[52] A. G. Salinger, N. M. Bou-Rabee, E. A. Burroughs, R. P. Pawlowski, R. B. Lehoucq, L. Romero, and E. D. Wilkes, Sandia Report SAND2002-0396, Sandia National Laboratories, 2002.
[53] J. Nocedal and S. Wright, "Quasi-Newton methods," *Numerical Optimization* (Springer, New York, 2006), pp. 135–163.
[54] R. Salazar and R. Toral, J. Stat. Phys. **89**, 1047 (1997).
[55] P. Ziherl and S. Svetina, Europhys. Lett. **70**, 690 (2005).
[56] A. M. Ferrenberg and R. H. Swendsen, Phys. Rev. Lett. **63**, 1195 (1989).
[57] U. Seifert, K. Berndl, and R. Lipowsky, Phys. Rev. A **44**, 1182 (1991).
[58] M. Jarić, U. Seifert, W. Wintz, and M. Wortis, Phys. Rev. E **52**, 6623 (1995).
[59] K. A. Brakke, Exp. Math. **1**, 141 (1992).
[60] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, SoftwareX **1–2**, 19 (2015).
[61] M. Hu, J. J. Briguglio, and M. Deserno, Biophys. J. **102**, 1403 (2012).
[62] See https://packaging.python.org/en/latest/guides/tool-recommendations/ for more information about standard tools to use with python.
[63] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, and E. Grinspun, Comput. Aided Geom. Des. **24**, 499 (2007).