On the bending algorithms for soft objects in flows

Achim Guckenberger[a,*], Marcel P. Schraml[a], Paul G. Chen[b], Marc Leonetti[c], Stephan Gekle[a]

[a] *Biofluid Simulation and Modeling, Fachbereich Physik, Universität Bayreuth, Bayreuth, Germany*
[b] *Aix-Marseille Université, CNRS, Centrale Marseille, M2P2, UMR7340, Marseille, France*
[c] *Aix-Marseille Université, CNRS, Centrale Marseille, IRPHE, UMR7342, Marseille, France*

Abstract

One of the most challenging aspects in the accurate simulation of three-dimensional soft objects such as vesicles or biological cells is the computation of membrane bending forces. The origin of this difficulty stems from the need to numerically evaluate a fourth order derivative on the discretized surface geometry. Here we investigate six different algorithms to compute membrane bending forces, including regularly used methods as well as novel ones. All are based on the same physical model (due to Canham and Helfrich) and start from a surface discretization with flat triangles. At the same time, they differ substantially in their numerical approach. We start by comparing the numerically obtained mean curvature, the Laplace-Beltrami operator of the mean curvature and finally the surface force density to analytical results for the discocyte resting shape of a red blood cell. We find that *none* of the considered algorithms converges to zero error at all nodes and that for some algorithms the error even diverges. There is furthermore a pronounced influence of the mesh structure: Discretizations with more irregular triangles and node connectivity present serious difficulties for most investigated methods.

To assess the behavior of the algorithms in a realistic physical application, we investigate the deformation of an initially spherical capsule in a linear shear flow at small Reynolds numbers. To exclude any influence of the flow solver, two conceptually very different solvers are employed: the Lattice-Boltzmann and the Boundary Integral Method. Despite the largely different quality of the bending algorithms when applied to the static red blood cell, we find that in the actual flow situation most algorithms give consistent results for both hydrodynamic solvers. Even so, a short review of earlier works reveals a wide scattering of reported results for, e.g., the Taylor deformation parameter.

Besides the presented application to biofluidic systems, the investigated algorithms are also of high relevance to the computer graphics and numerical mathematics communities.

Keywords: Helfrich Bending, Laplace-Beltrami Operator, Mean Curvature, Capsule Deformation in Shear Flow

1. Introduction

The computer simulation of soft deformable objects such as cells, synthetic capsules or vesicles in three-dimensional (3D) hydrodynamic flows is a rapidly increasing field in computational physics. The smallest investigated systems consider the dynamic motion of a single object in shear or channel flow [1–15], in a gravitational field [16–19], through narrow constrictions [20–22], or the diffusion of small particles near elastic membranes [23]. On a larger scale, a number of studies focus on the effective viscosity of dense suspensions [24–29] which is closely connected to the formation of cell-free layers near the channel walls in case of blood flow [30–33]. The investigation of suspensions containing two or more types of particles is another important field in which usually one focuses on the cross-streamline migration of the particles [30, 32, 34–41]. From a

*Corresponding author
Email address: achim.guckenberger@uni-bayreuth.de (Achim Guckenberger)

Accepted version. Published in *Computer Physics Communications* (2016), **207**, pp. 1–23. Doi: 10.1016/j.cpc.2016.04.018

computational perspective, an adequate method for the above problems requires two ingredients: Solution of a hydrodynamic problem for the flow for which a variety of methods such as Boundary Integral [42–44], Lattice-Boltzmann [28, 45–48] or particle methods [49–52] are available, and solution of a solid mechanics problem for the objects' interfaces.

The investigated objects are filled with fluid, separated from the outside by a membrane which is typically modeled as an infinitely thin elastic sheet. Forces originating from the linearized deformation of such a sheet can be split into in-plane elasticity (shear and area dilatation) and out-of-plane (bending) components. For the former a number of elastic laws such as neo-Hookean (e.g. [3, 53]) or Skalak [54] have been proposed, depending on the physical properties of the studied object, and recently different numerical modeling approaches have been compared [55]. Bending contributions are very often described via a simple law proposed by Canham and Helfrich [56, 57], stating that the local bending energy density is proportional to the square of the local mean curvature. Depending on the type of object, different contributions may dominate the total force. Vesicles, for example, lack shear elasticity and are thus entirely dominated by bending forces [5–7, 11, 58]. For elastic capsules, on the other hand, the elasticity governs most of the behavior, with bending causing mostly secondary effects [3, 28, 53, 58]. However, in certain situations it can become the dominating factor. For instance, it defines the wavelength of local wrinkles appearing for capsules especially at low shear rates due to local compressive forces [3, 53, 59–63]. Neglecting the bending rigidity in this case reduces not only the numerical stability but also the physical reliance greatly, making realistic simulations practically impossible [3, 28, 53, 61, 64]. For red blood cells, both elasticity and bending are relevant, where the latter determines the equilibrium shape [65]. Hence, the accuracy of the employed bending algorithm is of major concern.

To compute the mechanics of the membrane, it is typically discretized via a set of marker points whose positions are advected with the hydrodynamic flow. The most flexible, most easy-to-implement and therefore also one of the most widely used methods to interpolate between the nodes is a discretization via flat triangles [18, 43, 58, 63, 66–69]. Recently, subdivision surface methods [4, 70–74] are becoming increasingly popular, too. Other methods include curved triangles [2, 75, 76], B-Splines [3], or global approaches such as spherical harmonics [10, 20, 77]. The latter are most efficient for not too large deformations. Bending forces are computed as the derivative of the out-of-plane stress which, by the principle of virtual work, is the variational derivative of the bending energy [78, 79]. Since the mean curvature already contains the second derivative, in total the fourth derivative of the surface geometry is required. This poses a severe algorithmic and numerical challenge because the surface discretizations are often not $\mathcal{C}^4$ smooth.

Here we study a set of six algorithms to compute the bending forces for the most common case of a membrane discretized via flat triangles. A major difference between the algorithms is their approach on the Laplace-Beltrami operator, a key component of the bending forces. Note that its discretization is subject to active research [80–87]. For this work, we employ methods that are devised by or based on principles of Kantor and Nelson ([88], hereafter called Method A), Gompper and Kroll ([89], Method B), Meyer *et al.* ([90], Method C), Belkin *et al.* ([84], Method D), Farutin *et al.* ([68], Method E) and Loop and Cirak *et al.* ([73, 74], Method S). The latter, albeit being a subdivision method, also departs from flat triangles. To the best of our knowledge, no publication has so far used Belkin *et al.*'s discretization (Method D) for the computation of bending forces. In a recent work, Tsubota [69] compared three different algorithms akin to Methods A and C. He considered a shear flow setup and the equilibrium shape of a red blood cell (RBC), finding that Method A shows notable deviations to C. No comparison with an analytically solvable reference shape or earlier simulation work was attempted.

As a start we calculate the discretization error for the analytically known surface of an RBC. We find a strong difference in the quality and robustness of the algorithms: Most are very sensitive to the surface discretization and *none* converge at all nodes as the resolution is increased. The results are summarized in tables 2 and 3 (see page 17). To assess the performance of the bending algorithms in a typical flow setup, we then investigate the deformation of an initially spherical capsule in a viscous shear flow. The capsule is endowed with both shear and bending rigidity. To exclude any artifacts possibly arising from the flow solver, we use the Boundary Integral (BIM) as well as the Lattice-Boltzmann method (LBM). In general we find a very good agreement between both approaches. The deviations between the six bending algorithms are much less pronounced in this setup than in the analytical part. A comparison with the literature, however,

reveals a wide scattering of reported values for the Taylor deformation parameter.

We finally note that our study may also be relevant in other areas where the numerical evaluation of the Laplace-Beltrami operator, which is a main focus of this work, plays an important role. In geometry processing, for example, it is often used for the visualization of high-curvature regimes, highlighting of surface details, or surface smoothing and reconstruction [81, 84, 90, 91].

2. Computation of bending forces

2.1. The physical model of the bending energy

All bending algorithms used in the present work and in the majority of the literature depart from the seminal works of Canham [56] and Helfrich [57]. They considered a three-dimensional soft object with an infinitely thin interface endowed with bending resistance. They then proposed the following constitutive law for the bending energy:

$$E_\mathrm{B} = 2\kappa_\mathrm{B} \int_S [H(\boldsymbol{x})]^2 \, \mathrm{d}S(\boldsymbol{x}) \,. \tag{1}$$

Henceforth, S is the instantaneous smooth surface of the object and κ_B is the bending modulus. The local mean curvature is given by $H = \frac{1}{2}(c_1 + c_2)$, where c_1 and c_2 are the local principal curvatures. H is taken to be positive for a sphere. In principle an additional term dependent on the Gaussian curvature appears in the bending energy. Fortunately, this term is constant if the topology of the object does not change [57, 89]. Thus it is negligible for the purpose of force computations. A spontaneous (or reference) curvature can be included in equation (1) [65], but for simplicity we take the minimum energy reference state as a flat sheet.

For later convenience, we introduce an alternative expression for H [81]:

$$H(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{3} (\Delta_\mathrm{S} x_i) \, n_i(\boldsymbol{x}) \,, \quad \boldsymbol{x} \in S \,. \tag{2}$$

$\boldsymbol{n}(\boldsymbol{x})$ is the outer normal vector of the membrane surface S at position $\boldsymbol{x}$ and $\Delta_\mathrm{S} = \boldsymbol{\nabla}^\mathrm{S} \cdot \boldsymbol{\nabla}^\mathrm{S}$ denotes the Laplace-Beltrami operator with $\boldsymbol{\nabla}^\mathrm{S}$ being the surface gradient. Subscripts indicate vector components.

2.2. Principles for the computation of bending forces

The general goal is to compute the forces from the bending energy (1) while using an approximation for the surface S. As outlined in the introduction, we approximate S via flat triangles, i.e. each surface element consists of three nodes (vertices) and three straight edges. The force is then required at each node $\boldsymbol{x}^{(i)}$ with $i = 1, \ldots, N$. We denote by N the number of nodes and by N_T the number of triangles.

To be more precise, the hydrodynamic simulations performed in section 4 require either the force $\boldsymbol{F}^\mathrm{h}(\boldsymbol{x}^{(i)})$ (LBM) or the traction jump $\triangle \boldsymbol{f}^\mathrm{h}(\boldsymbol{x}^{(i)}) \coloneqq (\sigma_+ - \sigma_-) \cdot \boldsymbol{n}$ (BIM). σ_+ and σ_- are the stress tensors at the outside and inside of S, respectively, and $\boldsymbol{n}$ again the outer normal vector. The force equilibrium conditions read [53, 58]

$$\boldsymbol{F}^\mathrm{h} = -\boldsymbol{F} \quad \text{and} \tag{3}$$

$$\triangle \boldsymbol{f}^\mathrm{h} = -\boldsymbol{f} \,. \tag{4}$$

Depending on the employed bending algorithm, either the force $\boldsymbol{F}$ or the surface force density $\boldsymbol{f}$ is obtained. Conversion between both is thus necessary and will be described in section 2.6.

Computation of $\boldsymbol{F}$ or $\boldsymbol{f}$ means to perform a variational derivative of equation (1) with respect to small deformations of the surface. The six algorithms considered in this work (named A – E and S) effectively calculate this derivative and are summarized in table 1. For convenience, the online version of the article also allows to click on each occurrence of their names, enabling quick jumps to their respective chapters. From a conceptual standpoint, there is a major difference between the methods. The first two algorithms (Methods A and B) *first* discretize the surface and *then* perform the variational derivative by means of a direct differentiation with respect to the nodes' positions (termed "force formulation" below). This yields

Method	A	B	C	D	E	S
Basic ingredient from	Kantor [88]	Gompper [89]	Meyer [90]	Belkin [84]	Farutin [68]	Cirak [74]
Result	Force	Force	Force density	Force density	Force density	Force density
Derivative	Nodal	Nodal	Variational	Variational	Variational	FEM
Basic idea	Normal vector discretization	Δ_S via co-tangent scheme	Δ_S via co-tangent scheme	Δ_S via heat equation	Parabolic fitting	Subdivision surface

Table 1: Overview of the six methods A – E and S employed in this work.

the force $\boldsymbol{F}$. The three methods C – E perform the discretization *after* the variational derivative (termed "variational formulation" by us) and thus provide the force density $\boldsymbol{f}$. The last method S is similar to the force formulation as it introduces the discretization beforehand. However, it uses the weak formulation, solving a linear system of discretized integral equations by means of the finite element method (FEM). This leads again to the force density $\boldsymbol{f}$.

2.3. Force formulation

The first two Methods A and B first discretize the integral and the mean curvature from equation (1). The energy E_B then depends on the node coordinates $\boldsymbol{x}^{(i)}$ explicitly. By the principle of virtual work, they subsequently derive the force according to

$$\boldsymbol{F}(\boldsymbol{x}^{(i)}) = -\frac{\partial E_\mathrm{B}}{\partial \boldsymbol{x}^{(i)}}\,, \quad i = 1, \dots, N\,. \tag{5}$$

This derivative can often be performed analytically.

2.3.1. Method A

Method A starts with the expression $\int_S (H^2 - 2K)\,\mathrm{d}S = \int_S (\partial^\alpha \boldsymbol{n}) \cdot (\partial_\alpha \boldsymbol{n})\,\mathrm{d}S$ [89]. The integral with the Gaussian curvature K remains constant due to the Gauss-Bonnet theorem if the topology does not change and hence plays no role for the force calculation. The mean curvature part can be identified with equation (1). A direct discretization of the integral together with the approximation of equilateral triangles [89] then leads to the often employed expression (e.g. [9, 22, 27, 28, 32, 33, 36, 37, 40, 41, 52, 69, 88, 89, 92–95])

$$E_\mathrm{B} \approx 2\widetilde{\kappa}_\mathrm{B} \sum_{\langle i,j \rangle} (1 - \cos\theta_{ij})\,, \tag{6}$$

where the sum runs once over all edges $\langle i, j \rangle$. θ_{ij} is the angle between the normal vectors of the two triangles that contain edge $\langle i, j \rangle$.

One critical issue with this formula is the value of $\widetilde{\kappa}_\mathrm{B}$. It is usually not identical to the physical κ_B appearing in equation (1). In case of a sphere approximated by equilateral triangles it is simply $\widetilde{\kappa}_\mathrm{B} = \sqrt{3}\,\kappa_\mathrm{B}$. But in general the value depends on the shape of the object [89]. Nevertheless, one usually finds this value also being used for non-spherical shapes. Here we set $\widetilde{\kappa}_\mathrm{B} = \sqrt{3}\,\kappa_\mathrm{B}$, too. We remark that one could use equation (6) as the *model* equation directly, i.e. to take it not as an approximation of the Helfrich law in the first place. In this case our analysis must be viewed as addressing the question "how near or far away" it is from the Helfrich model rather than "how well of an approximation" it is.

A further simplification encountered from time to time is the usage of the small angle approximation $\cos\theta_{ij} \approx 1 - \frac{1}{2}\theta_{ij}^2$ [28, 29, 39, 52]. Its advantage is that it does not require the computation of a sine. We also tested this alternative and found a slight increase of the errors presented below. But because the error increase remained below 5 % and because the hydrodynamic simulations turned out to be insensitive to it, we restrict ourselves to the more common equation (6).

The analytic formulas for the derivative in equation (5) are presented in great detail in [28, ch. C.2] for the small angle approximation. Apart from the occurrence of an additional sine (stemming from the above

cosine), they can be also used unchanged for equation (6) and will therefore not be repeated here.[1] After knowing the force $\boldsymbol{F}$ at each node, the force density can be computed as outlined in section 2.6 below. This method is identical to "model KN" from Tsubota [69].

2.3.2. Method B

Gompper and Kroll [89] approximated the Laplace-Beltrami operator in the expression for the mean curvature from equation (2) by a variant of the so-called cotangent scheme, namely

$$\Delta_{\mathrm{S}} x_l^{(i)} \approx \frac{\sum_{j(i)} (\cot \vartheta_1^{(ij)} + \cot \vartheta_2^{(ij)})(x_l^{(i)} - x_l^{(j)})}{2A_{\mathrm{Voronoi}}^{(i)}}, \quad i = 1, \ldots, N, \quad l = 1, 2, 3, \tag{7}$$

where the sum runs over the first ring of neighbor nodes of $\boldsymbol{x}^{(i)}$. The integral from equation (1) is then discretized as

$$E_{\mathrm{B}} \approx \frac{\kappa_{\mathrm{B}}}{2} \sum_{i=1}^{N} \left(2H(\boldsymbol{x}^{(i)})\right)^2 A_{\mathrm{Voronoi}}^{(i)}, \tag{8}$$

with N denoting the total number of nodes. $\vartheta_1^{(ij)}$ and $\vartheta_2^{(ij)}$ are the angles opposite to the edge $\langle i, j \rangle$ in the triangles which contain nodes $\boldsymbol{x}^{(j-1)}$ and $\boldsymbol{x}^{(j+1)}$, respectively. See figure 1 for a sketch. $A_{\mathrm{Voronoi}}^{(i)}$ is the Voronoi area of node $\boldsymbol{x}^{(i)}$, defined by

$$A_{\mathrm{Voronoi}}^{(i)} := \frac{1}{8} \sum_{j(i)} (\cot \vartheta_1^{(ij)} + \cot \vartheta_2^{(ij)}) \left|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right|^2, \quad i = 1, \ldots, N. \tag{9}$$

The area is geometrically contoured by connecting the circumcenter points of the triangles. It must be noted that this Voronoi area does not lead to an exact covering of the surface if non-obtuse triangles occur [90], as outlined in the description of Method C below. Nevertheless, similar to Gompper and Kroll we use it for all triangles, no matter if obtuse or not.

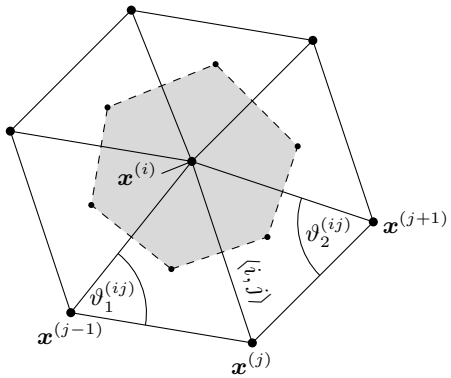


Figure 1: Ring-1 neighbors of some node $\boldsymbol{x}^{(i)}$. We marked one of the neighbors arbitrarily as $\boldsymbol{x}^{(j)}$. The shaded region depicts the Voronoi area $A_{\mathrm{Voronoi}}^{(i)}$.

The publication by Gompper and Kroll only provides the discretizations outlined above. To arrive at the force at node $\boldsymbol{x}^{(i)}$, we compute the gradient in equation (5) with the energy from equation (8) analytically, as explained in the Appendix A. Finally, the approach from section 2.6 is used to obtain the force density.

The solver presented in references [49, 50] and subsequent publications [96–100] is based on equation (8). "Model J" from Tsubota [69] is somewhat similar in spirit, but more simplistic.

[1] For the sake of completeness we note that for a non-zero reference state the formula $\theta_{ij} = \arccos(\boldsymbol{n}_i \cdot \boldsymbol{n}_j)$ as given in [28, ch. C.2] is only valid for convex parts of the surface. Otherwise, the value needs to be multiplied with -1 to correctly capture the reference shape. Here, $\boldsymbol{n}_i$ ($\boldsymbol{n}_j$) is the normal vector of the i'th (j'th) triangle. Furthermore, care must be taken if $\sqrt{1 - (\boldsymbol{n}_i \cdot \boldsymbol{n}_j)^2} \approx 0$, because then divisions through zero would occur. In this case the correct resulting force is simply zero for zero reference states. For non-zero reference states one needs to compute the correct value by an analytic limiting procedure.

2.4. Variational formulations

The three Methods C–E depart from an analytical and exact expression for the surface force density. It follows from the first variation of the bending energy $\delta E_\mathrm{B} = -\int_S \boldsymbol{f}\cdot\delta\boldsymbol{x}\,\mathrm{d}S$ as [8, 78, 79]

$$\boldsymbol{f}(\boldsymbol{x}) = 2\kappa_\mathrm{B}\left(2H(H^2-K)+\Delta_\mathrm{S}H\right)\boldsymbol{n}\,,\quad \boldsymbol{x}\in S\,. \tag{10}$$

Here, $K = c_1c_2$ is the Gaussian curvature with c_1 and c_2 being again the principal curvatures. $\boldsymbol{n}$ is the outer normal vector at $\boldsymbol{x}$ whose numerical computation also depends on the employed method. Similar expressions are obtained when a spontaneous or reference curvature is included [79]. The methods presented hereafter differ by how they compute the different quantities appearing in equation (10).

2.4.1. Method C

Meyer *et al.* [90] derive the following discretization for the Laplace-Beltrami operator on triangulated meshes from a contour integral around node $\boldsymbol{x}^{(i)}$ (also see references [58, 81]):

$$\Delta_\mathrm{S}w(\boldsymbol{x}^{(i)}) \approx \frac{\sum_{j(i)}(\cot\vartheta_1^{(ij)}+\cot\vartheta_2^{(ij)})(w(\boldsymbol{x}^{(i)})-w(\boldsymbol{x}^{(j)}))}{2A^{(i)}_\mathrm{mixed}}\,,\quad i=1,\dots,N\,. \tag{11}$$

w is an arbitrary two-times continuously differentiable function on S. The notation is otherwise identical to section 2.3.2. Obviously, it is another variant of the cotangent scheme. Comparing this equation with equation (7), we immediately see that they are almost the same with the sole difference being that the "mixed area" $A^{(i)}_\mathrm{mixed}$ rather than the Voronoi area $A^{(i)}_\mathrm{Voronoi}$ is used. For rings with non-obtuse triangles, one simply has $A^{(i)}_\mathrm{Voronoi} = A^{(i)}_\mathrm{mixed}$. However, using the Voronoi area for obtuse triangles leads to an incomplete tiling of the surface area, i.e. the sum of all Voronoi areas is not necessarily the same as the total surface area. For this reason, Meyer *et al.* introduced the mixed area: Rather than forming the area by all circumcenter points of all triangles, it uses the point in the middle of edges that are opposite of obtuse angles. Its precise definition and algorithm can be found in [90].

The mixed area aside, the second major difference compared to Method B is that equation (10) is being evaluated directly. Hence, after the mean curvature has been computed via equations (11) and (2), the Laplace-Beltrami operator of the mean curvature, $\Delta_\mathrm{S}H$, is calculated by applying formula (11) *again* on H [58].

For the Gaussian curvature K, Meyer *et al.* give the following simple expression [90]:

$$K(\boldsymbol{x}^{(i)}) = \frac{1}{A^{(i)}_\mathrm{mixed}}\left(2\pi-\sum_t\theta_t^{(i)}\right)\,,\quad i=1,\dots,N\,. \tag{12}$$

This is a discretization of the Gauss-Bonnet theorem. The sum runs over all triangles t sharing node $\boldsymbol{x}^{(i)}$, and $\theta_t^{(i)}$ is the angle in triangle t at node $\boldsymbol{x}^{(i)}$.

The last ingredient is the normal vector $\boldsymbol{n}(\boldsymbol{x}^{(i)})$. Several approaches exist to derive its value from the well-defined normal vectors of the triangles. Jin *et al.* [101] compared several often used methods. They concluded that summing the normal vectors of the triangles containing node $\boldsymbol{x}^{(i)}$ and weighting them with the angle $\theta_t^{(i)}$ gives the best results in many cases ("mean weighted by angle" approach, MWA). An often used alternative is to normalize the result of $\Delta_\mathrm{S}\boldsymbol{x}$. However, we found that the MWA algorithm provides superior results and thus we will adopt it in this work.

We finally remark that e.g. references [8, 12, 16, 18, 23, 35, 58, 66, 67, 80, 81, 102] and [44, 103–105] use the same or similar algorithms (where the latter refer to it as contour integral based method and only employ it for H). Method C is "model H" from Tsubota [69].

2.4.2. Method D

Method D differs from Method C by the discretization of the Laplace-Beltrami operator. It is based on a kernel of the diffusion or heat equation, and reads [84]

$$\Delta_{\mathrm{S}} w(\boldsymbol{x}^{(i)}) \approx \frac{1}{4\pi a_i^2} \sum_{t=1}^{N_{\mathrm{T}}} \frac{A_t}{3} \sum_{\boldsymbol{p}\in V(t)} \exp\left(-\frac{1}{4a_i}\left|\boldsymbol{p}-\boldsymbol{x}^{(i)}\right|^2\right)\left(w(\boldsymbol{x}^{(i)}) - w(\boldsymbol{p})\right), \quad i = 1,\ldots,N\,. \tag{13}$$

N_{T} denotes the number of triangles, $V(t)$ is the set of vertices of triangle t, and A_t its area. Furthermore, a_i is some free parameter that represents the neighborhood of node $\boldsymbol{x}^{(i)}$. Because it has the dimension of a squared length, we use $a_i = A^{(i)}_{\mathrm{mixed}}$ in the following. Other choices lead to very similar results.

Obviously, a single evaluation of the operator has a complexity of $\mathcal{O}(N_{\mathrm{T}}) \approx \mathcal{O}(N)$, where N is the number of nodes. Since we need to compute the bending forces at all vertices, Method D has an overall complexity of $\mathcal{O}(N^2)$ which can become prohibitively slow for larger meshes. On the other hand, the large supports leads to an insensitivity regarding noise [84], a fact which we also find reflected in our results below.

In practice we have slightly modified the above algorithm: Imagine a plane that goes through the centroid [106] of the object and with its normal vector pointing from the centroid to node $\boldsymbol{x}^{(i)}$. Then we only take into account points that lie *above* this plane. Otherwise, nodes that lie close in 3D space but are located far apart when measured along the surface (geodesic distance) would lead to large errors. This is the usual case for the dimples of red blood cells studied below.

Apart from the discretization of Δ_{S}, the remaining parts of the algorithm are identical to Method C. Especially note that the Gaussian curvature is still computed with Meyer *et al.*'s discretization given by equation (12) (because reference [84] does not provide an alternative), and the normal vector algorithm remains MWA. To the best of our knowledge, no publication so far used Belkin *et al.*'s discretization in the context of bending models.

It is also worth noticing that Li *et al.* [87] developed a similar formulation in a recent paper. Rather than using the *Euclidean* distance $\left|\boldsymbol{p}-\boldsymbol{x}^{(i)}\right|$ in the exponential function, they employ the *geodesic* distance between these points. Furthermore they do not take into account all triangles but only those within a certain cutoff (measured, again, via the geodesic distance). For most cases with mediocre resolution they report slightly better results for the mean curvature than for Belkin *et al.*'s discretization, becoming (mostly) better with increasing resolutions. Because of the only small advantage at practical resolutions and because it is non-trivial and often expensive to compute the geodesic distance between points on general triangular meshes [107], we have not yet attempted its implementation.

2.4.3. Method E

Instead of using a direct finite-differences like approach, Farutin *et al.* [68] employ a least square fitting procedure. The algorithm consists of three steps: In the first step, a local coordinate system at each node is created. Two axes are arbitrarily chosen to be parallel to the approximated tangential plane, while the third axis is parallel to the approximated normal vector. We estimate the normal vector again via the MWA algorithm. Next, one paraboloid is fitted to *each* of the three components of all the ring-1 nodes around each vertex. This involves the solution of three 5×5 linear systems per node. The fitting coefficients can be identified with local derivatives. They therefore provide a direct method to compute the mean and Gaussian curvature, the metric tensor and a refined approximation for the normal vector via standard differential geometry [68]. The final step fits a paraboloid to the mean curvature, yielding the coefficients required to evaluate $\Delta_{\mathrm{S}} H$. Putting everything together, the force density can then be computed via formula (10).

We remark that this algorithm requires each node to have at least five neighbors. Otherwise, the paraboloid would not be uniquely defined. For nodes with less than five neighbors, we extend the mean square fitting to include the ring-2. This leads to a well-defined problem for general meshes. Furthermore, in principle more rings can always be included as shortly discussed in section 3.5. An efficient algorithm for higher ring orders is breadth-first search.

Besides [68], publications that use the same basic idea to obtain the mean curvature include e.g. references [67, 81, 104, 108], although they fit a single paraboloid onto the surface itself (rather than to the components of the coordinates).

2.5. Method S

Method S is somewhat set apart from the previous five algorithms. Departing from the usual mesh with flat triangles, the subdivision scheme of Loop [73, 74] is applied to refine and average the surface. The method converges to a smooth limit surface which is $\mathcal{C}^2$ almost everywhere, corresponding to quartic box-splines. An exception are vertices that do not have six neighbors. There it is reduced to $\mathcal{C}^1$. With this method, the displacement field of an element depends on the (usually 12) neighboring triangles. In addition to being a versatile geometrical representation useful in computer aided design, the Loop subdivision is particularly well adapted to physical problems involving first and second-order derivatives such as infinitely thin shells described by a Kirchoff-Love energy functional. Indeed, the nodal forces at the membrane are determined by using the virtual work principle in its weak formulation, taking into account the membrane and bending strains [74]. This leads to a linear system of discretized integral equations that is treated with the finite element method. It means notably that neither Gaussian and mean curvatures nor the Laplace-Beltrami operator of the mean curvature need to be calculated explicitly. Still, the Gaussian and the mean curvature can be obtained from byproducts of the result, and we show them in section 3. We use GMRES [109] with a residuum of 10^{-9} to solve the linear system. The complexity in general is therefore $\mathcal{O}(N^2)$. This method has already been applied to capsules without bending resistance in a planar elongation flow [13] and droplets with dilational and shear surface viscosities [70] with a full validation section in each case. See reference [70] for further details.

References [4, 13, 70–72] also use similar subdivision surfaces in the biofluid context.

2.6. Conversion from forces to surface force densities

As already mentioned above, the quantity that is required to couple the membrane bending mechanics to the hydrodynamic flow depends on the employed flow solver. For our chosen LBM implementation one needs the force $\boldsymbol{F}$ at each surface node $\boldsymbol{x}^{(i)}$, while for BIM the surface force density $\boldsymbol{f}$ is required. By designating a certain area surrounding each node as the "node area" one can interconvert between both quantities by simply multiplying (dividing) the force density (force) by the respective node area [32, 69, 72]. This interconversion is necessary for a comparison of all six algorithms with a single flow solver, as Methods A and B yield the force $\boldsymbol{F}$ while C, D, E and S yield the force density $\boldsymbol{f}$.

We use Meyer *et al.*'s mixed area A_{mixed} already introduced in section 2.4.1 because of its perfect surface tiling property. Hence, the conversion is performed by the formula

$$\boldsymbol{f}(\boldsymbol{x}^{(i)}) \approx \frac{1}{A^{(i)}_{\text{mixed}}} \boldsymbol{F}(\boldsymbol{x}^{(i)}) \,, \quad i = 1, \ldots, N \,. \tag{14}$$

3. Benchmarking against analytical results for a static red blood cell

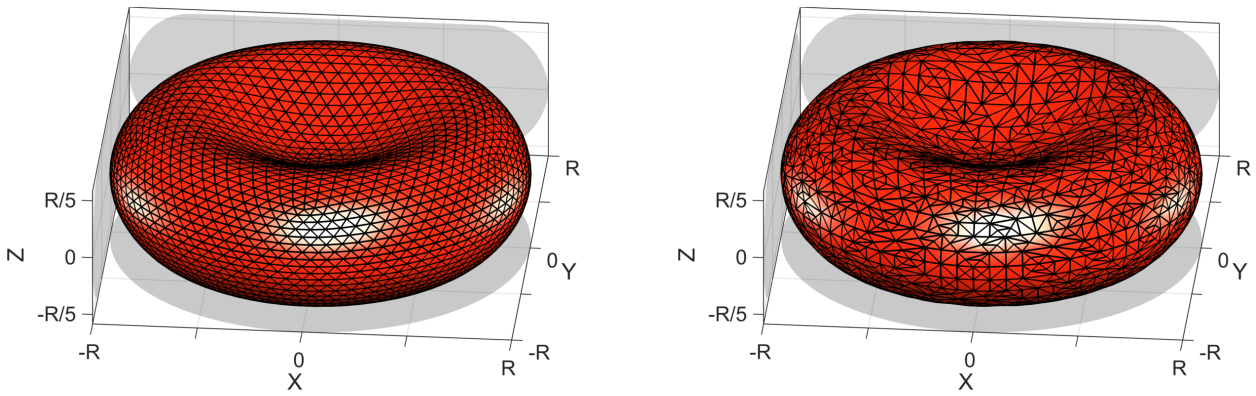
3.1. Red blood cell shape and methodology

RBC shape and discretizations. To quantitatively assess the quality and differences between the six bending approaches presented in the previous section, we consider the typical RBC shape as shown in figure 2. We choose this shape for two reasons: Firstly, it is simple enough to allow derivations of analytical expressions for all relevant quantities, including the force density itself, by means of differential geometry and standard computer algebra software. Secondly, it is a realistic shape which has regions where the mean curvature has different signs (i.e. turning points) and is thus complex enough to serve as a reasonable test subject. An oblate spheroid, for example, would be simpler to handle, but is a lot farther away from real-world situations in case of blood flow.

The considered shape can be described by the formula

$$z = \pm \frac{R}{2} \sqrt{1 - \rho^2} \left(C_0 + C_1 \rho^2 + C_2 \rho^4 \right) \tag{15}$$

with $\rho := \frac{1}{R}\sqrt{x^2 + y^2}$ and the constants $C_0 = 0.2072$, $C_1 = 2.0026$ and $C_2 = -1.1228$ [110, 111]. R is the length of the large half-axis of the RBC, and is often taken to be $R \approx 4\,\mu\text{m}$. In this section we use $R = 1$, effectively non-dimensionalizing all lengths by the RBC radius.



(a) Homogeneous mesh with 5120 triangles, obtained by refining an icosahedron using Loop's subdivision algorithm.

(b) Inhomogeneous mesh with 3914 triangles, obtained via Rivara's longest-edge bisection algorithm.

Figure 2: Illustrations of the typical red blood cell shape discretized with the two different MT1 methods as described in the main text. Meshes of type MT2 look very similar (see figure B.18 on page 29), except that the triangles around nodes with five neighbors are somewhat larger.

The discretization as shown in figure 2a is derived by successively refining a regular icosahedron via division of each triangle into four new elements according to Loop's subdivision scheme [73, 74]. The z-coordinates of the nodes are afterwards modified via application of formula (15). This leads to the very homogeneous meshes with 320, 1280, 5120, 20480 and 81920 triangles considered below. All of the nodes have six neighbors, with the exception of exactly twelve vertices retaining only five neighbors for any N. The meshes with 512 and 2048 triangles are based on a regular octahedron, leading to four or six neighbors. We call discretizations based on Loop's refinement "MT1".

We also assess the behavior on three other mesh types for Methods A – E. The first is the inhomogeneous one shown in figure 2b. This mesh with 3914 triangles is obtained by starting from an icosahedron refined to 320 triangles via Loop's scheme (MT1). We then apply Rivara's longest-edge bisection algorithm [112] three times and transform the result via equation (15) to the RBC. Each node has four to ten neighbors. The second additional mesh is very similar to the homogeneous geometries described above, except that new nodes introduced during the refinement are simply placed at the middle of edges rather than according to Loop's algorithm. This is an often used scheme, and results in a slightly different structure. We will refer to it as "MT2" and outline observed differences in the main text where appropriate. The actual data can be found in Appendix B. As a third mesh we consider the application of Rivara's algorithm to an MT2 object with 320 elements, resulting in 3848 triangles. See the supplementary information (SI) for a collection of mesh properties such as typical edge lengths.

Evaluation and error measures. All numerical results were obtained using double precision arithmetic. They are plotted as functions of the polar angle θ_i, computed via

$$\theta_i = \arccos\left(\frac{z_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}\right), \tag{16}$$

where x_i, y_i and z_i are the Cartesian coordinates of node $\boldsymbol{x}^{(i)}$, $i = 1, \ldots, N$. The RBC described by equation (15) is axisymmetric and hence the results at each position will be plotted as a function of the polar angle only. This corresponds to a projection of the azimuthal direction into a single plane. We do not just compute the data at some cross section because the *mesh* is non-axisymmetric and hence results do vary with the azimuthal angle.

The relative errors of the normal vector, the mean curvature and the force density are computed as

normalized Euclidean norms:

$$\varepsilon_{\boldsymbol{n}}(\theta_i) \coloneqq |\boldsymbol{n}^{\mathrm{a}}(\boldsymbol{x}^{(i)}) - \boldsymbol{n}^{\mathrm{n}}(\boldsymbol{x}^{(i)})|\,, \quad \varepsilon_H(\theta_i) \coloneqq \frac{|H^{\mathrm{a}}(\boldsymbol{x}^{(i)}) - H^{\mathrm{n}}(\boldsymbol{x}^{(i)})|}{\max|H^{\mathrm{a}}|}\,, \quad \varepsilon_{\boldsymbol{f}}(\theta_i) \coloneqq \frac{|\boldsymbol{f}^{\mathrm{a}}(\boldsymbol{x}^{(i)}) - \boldsymbol{f}^{\mathrm{n}}(\boldsymbol{x}^{(i)})|}{\max|\boldsymbol{f}^{\mathrm{a}}|}\,, \quad (17)$$

where the superscripts a and n denote the analytical and the numerical value, respectively. Errors for the Gaussian curvature, ε_K, and the Laplace-Beltrami operator of H, $\varepsilon_{\Delta_{\mathrm{S}}H}$, are handled the same way as ε_H. Note that $\varepsilon_{\boldsymbol{n}}$ is the error relative to the length of the normal vector $|\boldsymbol{n}^{\mathrm{a}}| = 1$. Furthermore, we compute the errors relative to the maximal analytic values because transitions through zero exist. Using the local analytic results as the reference would lead to greatly exaggerated error values: The numeric algorithms never produce zero values at precisely the same positions as the analytics. Approximately, one has for $R = 1$ and $\kappa_{\mathrm{B}} = 1$: $\max|H^{\mathrm{a}}| \approx 2.20098$, $\max|K^{\mathrm{a}}| \approx 3.60620$, $\max|\Delta_{\mathrm{S}}H^{\mathrm{a}}| \approx 109.095$ and $\max|\boldsymbol{f}^{\mathrm{a}}| \approx 189.457$.

Continuing, the tables and graphics presented in this work show two different measures for the total error. The maximum error of all the nodes is calculated as $\max_i \varepsilon_\bullet(\theta_i)$ and represents the local worst case result. On the other hand, the average error is computed as $\sum_{i=1}^{N} \varepsilon_\bullet(\theta_i)/N$ with N being the number of nodes on the RBC. It measures the overall performance of the algorithm for viscous flows. More precisely speaking, the average can be interpreted as a discretization of a continuous error measure, for example $\sum_{i=1}^{N} \varepsilon_{\boldsymbol{f}}(\theta_i)/N \sim \int_S |\boldsymbol{f} - \boldsymbol{f}^{\mathrm{a}}|\,\mathrm{d}S$, because $1/N \sim h^2 \sim \mathrm{d}S$ where h is the mean edge length. Similar integrals determine the flow in the Stokes regime, compare section 4.3.1. The average measure is therefore the more meaningful quantity in this particular application. Indeed, previous studies reported that a few problematic nodes do not affect the overall hydrodynamic results significantly [68]. We confirm this in section 4.

Below we show the numerical results as a function of θ. Corresponding images of the errors can be found in the supplementary material. Furthermore, we display the two error measures (maximum and average) as a function of the inverse of the *mean* edge length h or, equivalently, as a function of triangle count N_{T}.

We start by considering the normal vector, the mean curvature and the Laplace-Beltrami operator of the mean curvature. These are required for the evaluation of equation (10) in the variational formulation. The Gaussian curvature is considered in the supplementary information. As a last step we combine the results to get the force density $\boldsymbol{f}$, where we also include the two algorithms based on the force formulation. Tables 2 and 3 at the end of this section summarize all results.

3.2. Normal vector

As outlined in section 2.4, we use the MWA algorithm to compute the normal vector for Methods C and D. In case of Method E, the MWA normal vector is taken as the input and the fit procedure yields a new result. Farutin *et al.* [68] stated a convergence rate for E of $\mathcal{O}(h^2)$.

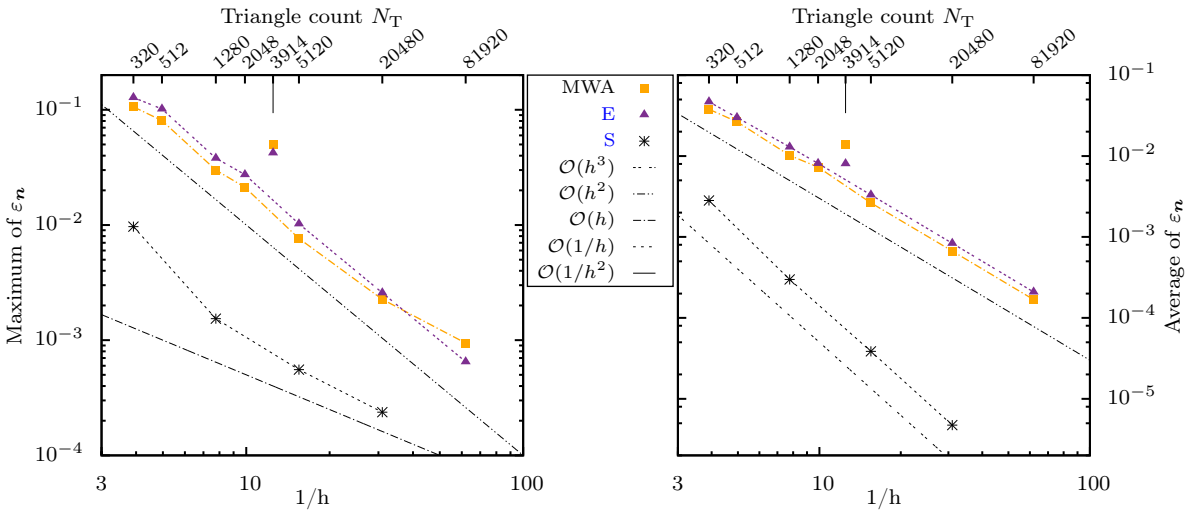
Figure 3 shows the maximum and average of the relative error $\varepsilon_{\boldsymbol{n}}$ of the normal vector for the MT1 based meshes. Obviously, E as well as MWA converge roughly with $\mathcal{O}(h^2)$ in both error measures at the beginning. For E this is in line with the report of Farutin *et al.* At larger resolutions MWA appears to decay as $\mathcal{O}(h)$ for the maximum error. We remark that the computation of $\boldsymbol{n}$ by normalization of $\Delta_{\mathrm{S}}\boldsymbol{x}$ for Methods C and D often leads to errors that are an order of magnitude larger.

The subdivision based Method S is a bit different. Its maximal error tends to behave first as $\mathcal{O}(h^2)$ and then as $\mathcal{O}(h)$, similar to MWA, although the absolute values are roughly one order of magnitude smaller. The major error source are nodes that are members of the ring-1 neighborhood of vertices that have only five neighbors, i.e. where the surface is only $\mathcal{C}^1$ smooth. On the other hand, the majority of the surface is $\mathcal{C}^2$ which leads to the observed $\mathcal{O}(h^3)$ convergence of the average error.

Considering the inhomogeneous mesh with 3914 triangles, both MWA and Method E are found to be sensitive to irregularities, albeit E is a bit less affected. Furthermore, both show similar behavior on the second mesh type MT2, except that MWA decays as $\mathcal{O}(h)$ starting with 5120 elements (see figure B.19 in Appendix B).

3.3. Mean curvature

The mean curvature H is one of the central ingredients for the computation of the bending energy (cf. equation (1)) and, subsequently, the bending forces. However, it is explicitly required only for Methods C,



(a) Maximum of all nodes.

(b) Average over all nodes.

Figure 3: The maximum and average of the relative error $\varepsilon_{\boldsymbol{n}}$ of the normal vector as a function of resolution N_T or, equivalently, as a function of the inverse of the mean edge length h for the RBC shape MT1. MWA is the "mean weighted by angle" algorithm from [101]. Black lines without symbols depict typical scaling behaviors. Results for the inhomogeneous mesh with 3914 triangles are highlighted by the small vertical line at the top. The numerical values and a breakdown per node as a function of the polar angle θ can be found in the supplementary material. The results for the MT2 mesh types are in figure B.19.

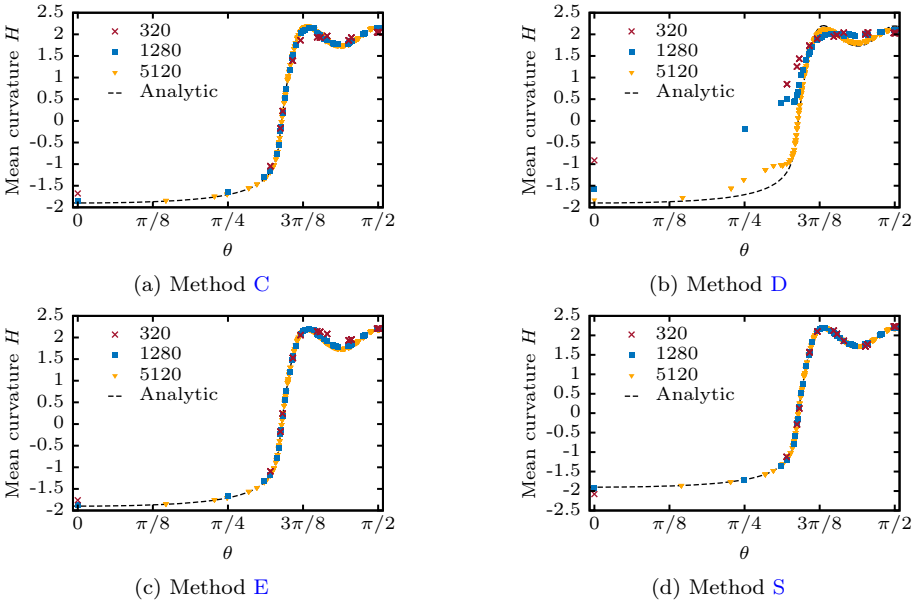
D and E. Method S yields it as a byproduct. The mean curvature from these four approaches is shown in figure 4 in comparison to the analytical result. Correspondingly, figure 5 presents the maximal and average errors.

Method C seems to retain a systematic maximal error because the results at a few nodes slightly above $\theta = 3\pi/8$ do not converge to the correct result (identical to the highlighted nodes in figure 9c on page 15). These nodes have six neighbors, just like most of the other vertices and thus are not very "special" at first sight. For further analysis we turn to Xu [80] who gives a *sufficient* condition for convergence: Firstly, the node must have six neighbors. Secondly, there must exist a local parametric representation $q \in \mathbb{R}^2$ of the surface such that $q_j = q_{j-1} + q_{j+1} - q_i$ where i indicates some node and j enumerates its six neighbors. If we choose at each node a coordinate system with two axes in the tangential plane and the last axis along the (analytic) normal vector, we find that this condition is most severely violated at precisely the non-converging points. This suggests why we find a systematic error. Note, however, that this is merely a sufficient rather than a necessary condition for convergence. As such, convergence can and is observed at other nodes although they also violate this condition or do not even have six neighbors. The same is observed for the MT2 meshes (figure B.20 in Appendix B).

Both Method D's and E's approaches lead to convergent results regarding the maximum error. The rate is roughly $\mathcal{O}(h)$ for Method E which is consistent with reference [68]. Hence we find that algorithm E is superior to C as also noticed by Zinchenko *et al.* [104] for two variants of the methods. For D the rate seems to be $\approx \mathcal{O}(h^2)$, albeit it has error levels that are typically half to one order of magnitude larger than E for the present case. Notice that D's operator can actually be proven to converge point-wise for arbitrary meshes and hence also in the here employed Euclidean norm [84], a fact which is reflected in our results. Method S has problems with the $\mathcal{C}^1$ regions as error levels quickly stagnate. The absolute values are still comparably small.

With respect to the average errors, algorithms C – E display the same behavior: they converge as $\mathcal{O}(h^2)$. Method D is still the one with the largest errors. E and C are alike, with E having a slight edge. The subdivision scheme S appears to converge faster at first. At higher resolutions, the rate reduces to $\mathcal{O}(h^2)$.

Finally, we consider the inhomogeneous mesh with 3914 triangles. Algorithm E shows a serious increase



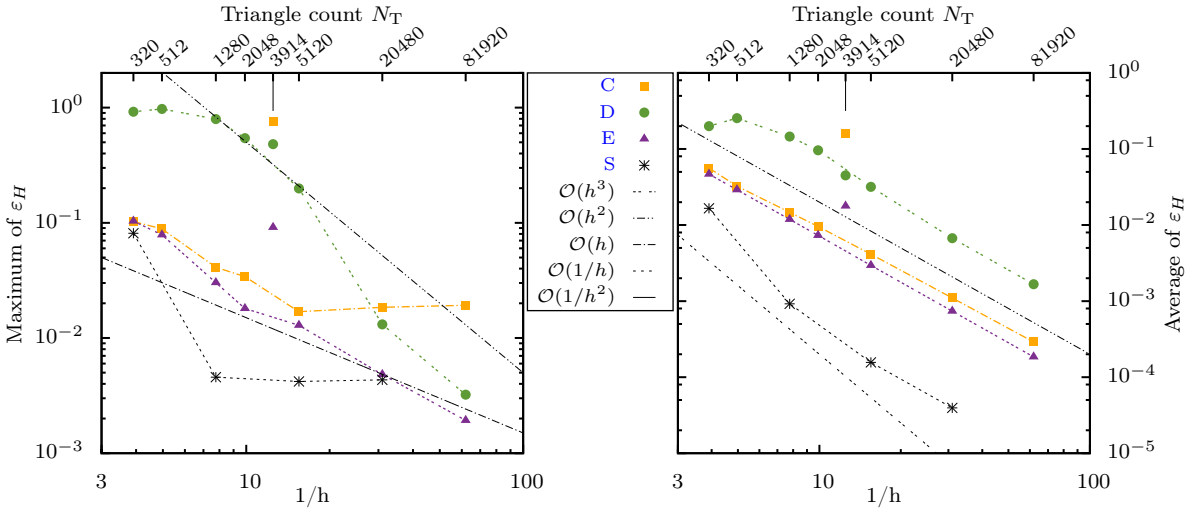
(a) Method C

(b) Method D

(c) Method E

(d) Method S

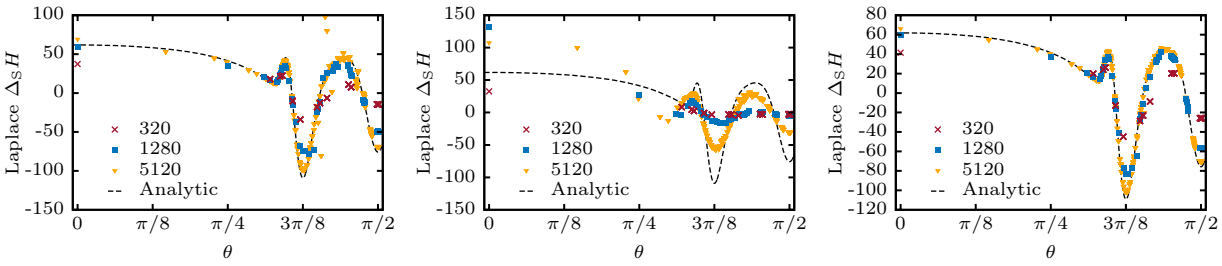
Figure 4: The mean curvature H of a MT1 RBC obtained from the four different algorithms (symbols) and compared to the analytical result (dashed line) as a function of the polar angle θ. The errors at each node can be found in the supplementary information.



(a) Maximum of all nodes.

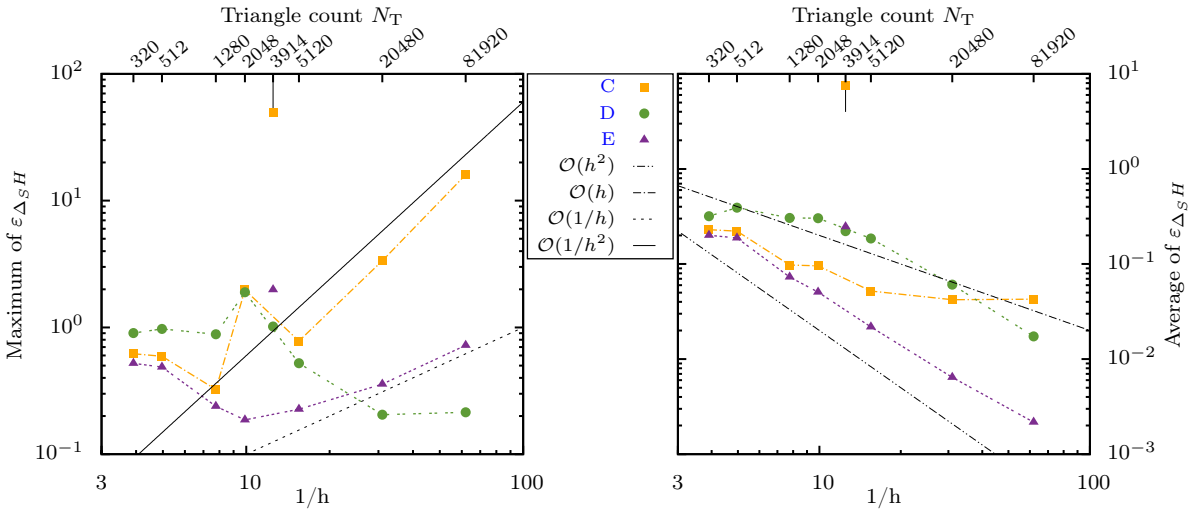
(b) Average over all nodes.

Figure 5: The maximum and average of the relative error ε_H of the mean curvature (MT1). For the numeric values, see the supplementary information. The results for MT2 are in figure B.20.



(a) Method C (b) Method D (c) Method E

Figure 6: The result of evaluating the Laplace-Beltrami operator on the mean curvature, $\Delta_S H$. MT1 meshes. Figures with the corresponding errors can be found in the supplementary information.



(a) Maximum of all nodes. (b) Average over all nodes.

Figure 7: The maximum and average of the relative error of the Laplace-Beltrami operator applied to H. MT1 meshes. The results for MT2 are in figure B.21. The numeric values are in the supplementary information.

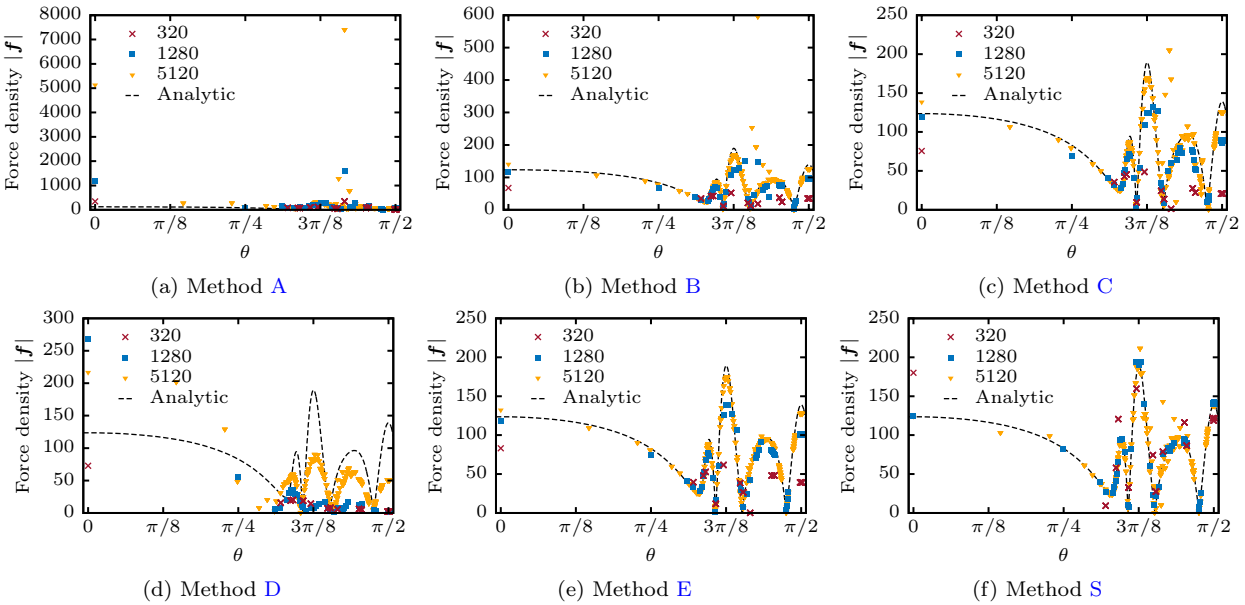
of the error while it is tremendous for Method C. The largest errors for C come from nodes with only four neighbors, although some of them show smaller deviations (also note that the meshes with $N_T = 512$ and $N_T = 2048$ have vertices with four adjacent nodes). On the other hand, Method D is largely unaffected by the irregularities. This indicates that it is more robust than the other algorithms, an observation which can be attributed to the larger support [84].

We remark that the MT2 meshes lead to the same conclusions, except that the error magnitudes can be significantly larger (cf. Appendix B). The Gaussian curvature is analyzed in the supplementary material, revealing similar results.

3.4. Laplace-Beltrami operator of the mean curvature

The remaining quantity that needs to be computed for equation (10) is the Laplace-Beltrami operator of the mean curvature, $\Delta_S H$. As described in section 2.4, it is calculated by applying the Laplace discretization idea again to H for C, D and E. Method S does not easily allow the computation of $\Delta_S H$. Considering the results from the previous section 3.3, we do not expect Method C to converge. Our findings are shown in figures 6 and 7.

Obviously, the expectation is met: Method C fails to converge entirely. The maximum of the error even diverges quickly as $\mathcal{O}(1/h^2)$ because of the troublesome points already identified for the mean curvature.



(a) Method A (b) Method B (c) Method C

(d) Method D (e) Method E (f) Method S

Figure 8: The magnitude of the force density $\boldsymbol{f}$ computed via the six different bending algorithms and compared to the analytical result. MT1 meshes and $\hat{\kappa}_\mathrm{B} = 1$. The errors are displayed in figure 9.

Algorithm E at first appears to become more precise with higher resolutions in the maximum error measure, but then appears to diverge with a rate of $\approx \mathcal{O}(1/h)$ for $N_\mathrm{T} \geqslant 5120$. Indeed, Farutin *et al.* report the same trend. In the context of Stokes flow, however, they found that it can still provide very good results [68] (also see the average error and section 4). Even the deviations for the rather lenient Method D fail to decay with resolutions beyond 20480 triangles.

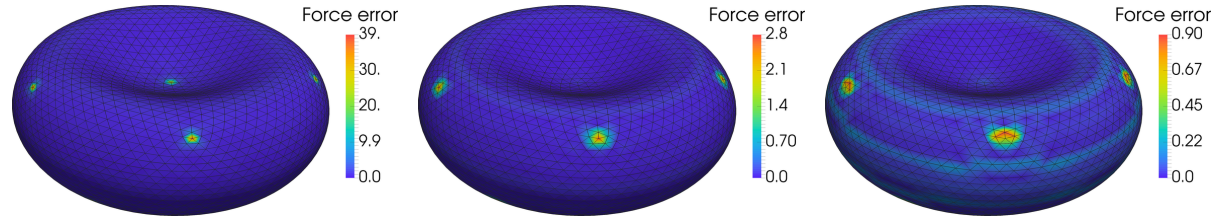
Regarding the average error measure, Method C retains a systematic deviation. Methods D and E both converge with roughly $\mathcal{O}(h^2)$ in the presented case. Unfortunately, for meshes of type MT2 we observe an entirely different behavior (see figure B.21 in Appendix B): The rate for D is reduced to $\mathcal{O}(h)$, and E fails to converge (although it does not diverge). Hence, D is again much more robust with respect to different triangulations than E.

This is also reflected in the results for the inhomogeneous mesh with 3914 triangles (see figure 7). Method E and especially C depict severe difficulties for both the maximal and average errors, yielding values that are mostly one to two orders of magnitude larger than for 1280 triangles. In contrast to the previous sections, Method D also shows some problems for the maximal error, although it can keep the average error on an adequate level.

3.5. Force density

Finally, we combine all the quantities via equation (10) and get the force density $\boldsymbol{f}$ for Methods C, D and E. Additionally, Methods A, B and S are applied as described in sections 2.3, 2.5 and 2.6. We plot the results in figures 8, 9 and 10. The bending modulus has been set to $\kappa_\mathrm{B} = 1$ in all cases.

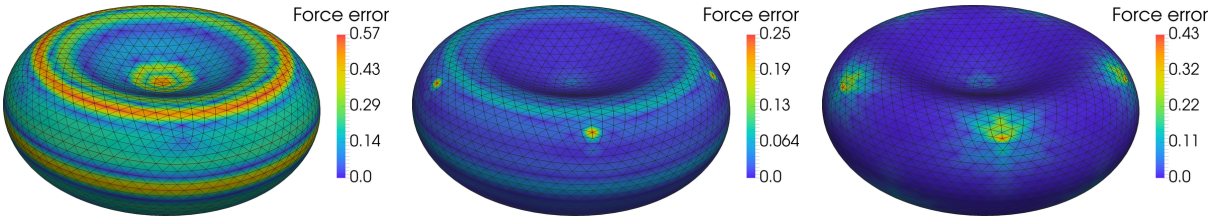
Variational formulation. Since the value of $\Delta_\mathrm{S} H$ dominates equation (10), it is of no surprise that the errors $\varepsilon_{\boldsymbol{f}}$ for C, D and E are very similar to $\varepsilon_{\Delta_\mathrm{S} H}$. Notably, Method D appears to converge with $\mathcal{O}(h)$ only for the MT2 average error measure but not for the maximal, retaining a systematic deviation there. The maximal errors for E and C diverge with $\mathcal{O}(1/h)$ and $\mathcal{O}(1/h^2)$, respectively. The averaged error remains roughly constant for C, but decays as $\mathcal{O}(h^2)$ for E. As noted above, this last rate does not carry over to the MT2 meshes in Appendix B (figure B.23) where E behaves similar to C. To this end, the error pattern on the surface differs notably between MT1 (figure 9e) and MT2 (figure B.22e).



(a) Method A: Nodes without six neighbors are troublesome (also on other meshes).

(b) Method B: Compared to C other nodes cause problems. In this particular case, the nodes with five neighbors.

(c) Method C: The problematic nodes here are not related to five neighbor regions and are analyzed in section 3.3.

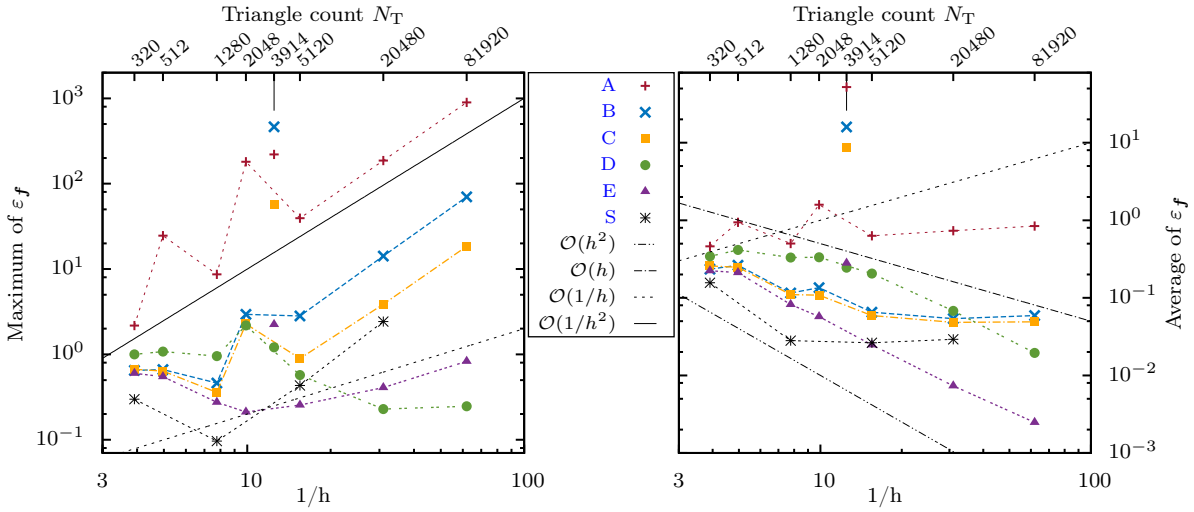


(d) Method D: The circular patterns indicate an independence of the errors of the mesh topology.

(e) Method E: A circular pattern similar to Method D appears. Inclusion of more neighbor-rings leads to more pronounced circular patterns (not shown).

(f) Method S: Problems at nodes with only five neighbors occur, i.e. where the surface is only C^1 smooth.

Figure 9: 3D illustration of the errors ε_f of the force density from figure 8 for 5120 triangles. MT1 meshes. The results for MT2 are in figure B.22. Note the scales of the color bars. A projection of the azimuthal direction onto a single plane can be found in the supplementary information.



(a) Maximum of all nodes.

(b) Average over all nodes.

Figure 10: The maximum and average of the relative error of the force density. MT1 meshes. The results for MT2 are in figure B.23. For the numerical values see the supplementary information.

Force formulation. Regarding the force formulation, Method B is alike to C regarding the general behavior. However, the latter has a notable edge over the first. C is also less (but still a lot) affected by more irregular triangulations. The same holds for MT2 meshes, too. There are only two possible sources for these deviations, since both use virtually the same discretization of the Laplace-Beltrami operator as outlined in section 2.4.1. The first reason could be the imperfect surface tiling used for Method B (Voronoi rather than mixed area). Secondly, the differing approach for the variational derivative (before and after discretization). A quick check where we replaced $A^{(i)}_{\text{mixed}}$ with $A^{(i)}_{\text{Voronoi}}$ in Method C (equation (11)) leads to almost identical values as B (except for the inhomogeneous mesh where the errors are even more extreme). Therefore we conclude that at least for meshes with mediocre homogeneity the choice of either force or variational formulation has only a minor influence, in agreement with [69]. On the other hand, a proper surface tiling turns out to be important. Nevertheless, we remark again that both methods still exhibit very high sensitivity to the mesh regularity, and both do not converge.

Method A always shows extreme errors: Not only are the absolute values orders of magnitude larger than for the other methods and its maximal error diverges as $\mathcal{O}(1/h^2)$, but even the average error increases with the resolution at a rate of $\mathcal{O}(1/h)$ for MT2 meshes (see figure B.23 in Appendix B). The major problems originate from nodes with a different number of neighbors than six (see figure 9a).

Method S. The subdivision surface algorithm, Method S, exhibits a $\mathcal{O}(1/h^2)$ divergence of the maximal error, just like A, B and C. The average error also saturates, although with a significantly smaller value than the others. As noted before and as clearly visible in figure 9f, the five neighbor regions contain the biggest troublemakers, most likely because the surface is only $\mathcal{C}^1$ smooth there.

Method E with a larger support. We also tested a slight modification of Method E that takes into account not just the first ring of neighbors for the fitting procedure, but also the second and third rings. The values are included in the tables in the supplementary material. In general we found three differences: First of all, for smaller resolutions ($N_{\text{T}} \leqslant 5120$) the errors are larger for higher ring orders. Second, the start of the $\mathcal{O}(1/h)$ section in the maximal error measure is postponed to higher resolutions, but it still exists. Third, they can handle the inhomogeneous mesh far better but still not as good as approach D. Therefore, we conclude that taking into account more neighbors benefits the overall robustness of the fitting method (as reported in [82]), albeit leading to larger errors at mediocre (and thus often practical) resolutions. This is in line with Method D, which is an extreme case, having all nodes as support. Indeed, the 3D error pattern of E (figure 9e) becomes more similar to that of algorithm D (figure 9d). But note that the underlying discretization idea of the latter is completely different.

Final notes. One also has to emphasize the absolute value of the errors: No method yields relative maximal errors below 10 % for all considered resolution, and they are often in the $> 200\,\%$ regime for irregular meshes. This illustrates the basic difficulty of computing a fourth order derivative on triangulated meshes. We remark that it is theoretically impossible to construct a discrete Laplace-Beltrami operator that satisfies all expected properties simultaneously [83].

A concise summary of all the results can be found in tables 2 and 3 using the MT2 meshes for Methods A – E (MT1 is the same or better) and MT1 for Method S. Due to the revealed mesh dependencies, the exact quality of the algorithms highly depends on the individual case. Still we are positive that the tables can serve as a general guideline.

4. An elastic capsule in shear flow

4.1. Basic setup

We now analyze the performance of the bending algorithms in the context of Stokes flow. Despite the sometimes very large deviations compared to the analytics as presented in the previous section, one can expect a better performance in such an application [68, 69]. As already noted above, viscous flows lead to an averaging of the errors.

Method	A	B	C	D	E	S
K			○	○	●	○
H			○	●	●	○
$\Delta_S H$			■	○	■	
$\boldsymbol{f}$	■	■	■	○	■	■

(a) Maximal errors

Method	A	B	C	D	E	S
K			●	●	●	●
H			●	●	●	●
$\Delta_S H$			○	●	○	
$\boldsymbol{f}$	■	○	○	●	○	○

(b) Average errors

Table 2: Summary of the convergence results for the RBC. The symbol ● indicates convergence to zero errors, ○ convergence with a systematic error and ■ divergence as the resolution is increased. K is the Gaussian curvature, H the mean curvature and $\boldsymbol{f}$ the force density. Note that only in Method D the average errors of the force density seem to converge to zero, while no algorithm shows convergence of the maximum error.

Method	A	B	C	D	E
K			■	■	■
H			■	●	■
$\Delta_S H$			■	○	■
$\boldsymbol{f}$	■	■	■	○	■

Table 3: Summary of problematic behavior for the inhomogeneous mesh. ● means no troubles, ○ labels problems only in the maximal but not in the average error measure and ■ indicates a significant increase of the deviations in both error measures.

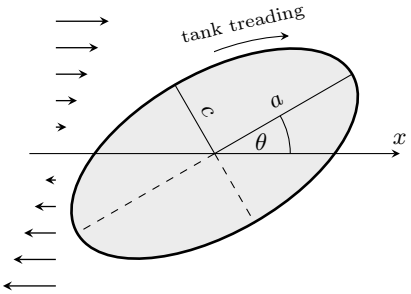


Figure 11: 2D illustration of the linear shear system. The actual simulation is three-dimensional. The spherical capsule deforms to an ellipsoid-like shape. a is the largest and c the smallest half axis, while θ is the inclination angle. The membrane rotates around the object's centroid (the so-called "tank-treading" motion).

We consider an initially spherical capsule of radius R placed in a linear shear flow with shear rate $\dot{\gamma}$ as illustrated in figure 11. The Reynolds number is much smaller than one. This is an often studied system and performance test case in the recent literature [1–4, 8, 10, 55, 62, 69, 71, 75, 102, 113, 114], also motivated by several experiments that displayed varying results [14, 59, 115]. The infinitely thin capsule surface is endowed not just with a bending rigidity, but additionally with a shear elasticity as described in the next section 4.2. Its inside is filled with a Newtonian fluid having the same viscosity as the ambient flow. It is well known that for not too large shear rates the shape of such an object becomes approximately an ellipsoid. This state is usually described by the Taylor deformation parameter $D := \frac{a-c}{a+c}$ with the largest and smallest semi axes a and c, respectively, and the inclination angle θ between the x-axis and a. We extract D and θ at each time step from an ellipsoid with the same inertia tensor [48, 116].

The system parameters can be cast into two dimensionless values [113]: the dimensionless shear rate (or elastic capillary number) $G := \frac{\mu\dot{\gamma}R}{\kappa_S}$ and the dimensionless ratio between shear and bending resistance $\hat{\kappa}_B := \frac{\kappa_B}{R^2\kappa_S}$. κ_S is the shear modulus for the in-plane tensions (compare the next section). Note that different conventions exist in the literature. Furthermore, the reference state for these in-plane tensions is taken to be the initial sphere, whereas the bending reference state is a flat sheet (also see section 2). Numerically, the sphere is constructed using Loop's subdivision as presented in section 3.1 (MT1), just without the final transformation to the RBC shape. We consider 320, 1280 and 5120 triangles as well as the inhomogeneous 3914 triangle mesh. The results with MT2 meshes are practically identical and will therefore not be discussed any further.

4.2. In-plane forces

Apart from the bending forces, we additionally take into account elastic in-plane tensions to allow for comparisons with the existing literature. We choose the widely used neo-Hookean law (e.g. [3, 53]) whose

in-plane energy density due to stretching can be written as

$$\epsilon_{\mathrm{S}} = \frac{\kappa_{\mathrm{S}}}{6}\left(I_1 + \frac{1}{I_2+1} - 1\right) . \tag{18}$$

Other forms exist in the literature. I_1 and I_2 are the strain invariants. They are related to the principal in-plane stretch ratios λ_1 and λ_2 via

$$I_1 = \lambda_1^2 + \lambda_2^2 - 2 \quad \text{and} \tag{19a}$$
$$I_2 = \lambda_1^2\lambda_2^2 - 1. \tag{19b}$$

The total energy is given by the surface integration

$$E_{\mathrm{S}} = \oint_{S_0} \epsilon_{\mathrm{S}} \,\mathrm{d}S_0 \,, \tag{20}$$

where the surface S_0 denotes the surface in the *reference* (i.e. initial) state.

Assuming that the deformation varies linearly over the triangles, the force is then obtained via the finite element method by performing the derivative of equation (20) with respect to the node positions $\boldsymbol{x}^{(i)}$

$$\boldsymbol{F}(\boldsymbol{x}^{(i)}) = -\frac{\partial E_{\mathrm{S}}}{\partial \boldsymbol{x}^{(i)}} \tag{21}$$

analytically, just like it was done for the bending forces in section 2.3. Section 2.6 is used to arrive at the force density. The details are elaborated in references [1, 28] and will not be repeated here. We remark that this method gives very good results when compared with the literature, as shown below in section 4.4.

4.3. Flow solvers

We now describe the two employed flow solvers: The Boundary Integral method (BIM) and the Immersed-Boundary Lattice-Boltzmann method (LBM). As the main focus of this article is on the computation of bending forces, we will keep the description fairly brief.

4.3.1. Boundary Integral Method

The Boundary Integral method was first applied in the context of Stokes flow by Youngren and Acrivos in 1975 [42]. Its basic assumption is that the Reynolds number is much smaller than unity. The method then exploits the fact that the Stokes equation is linear, and can therefore be rewritten as an integral equation [43, 44]:

$$u_j(\boldsymbol{x}) = u_j^{\infty}(\boldsymbol{x}) - \frac{1}{8\pi\mu}\oint_S \sum_{i=1}^{3} \triangle f_i^{\mathrm{h}}(\boldsymbol{y}) G_{ij}(\boldsymbol{y},\boldsymbol{x}) \,\mathrm{d}S(\boldsymbol{y}) \,, \quad \boldsymbol{x} \in S \,, \quad j = 1,2,3 \,. \tag{22}$$

$\boldsymbol{x}$ is a point on the surface S of the capsule that is suspended in the infinite fluid domain, $\boldsymbol{u}$ is the surface velocity and μ the dynamic viscosity of the inner and outer fluids. $\boldsymbol{u}^{\infty}(\boldsymbol{x}) = \dot{\gamma} x_3 \hat{\boldsymbol{e}}_x$ is the imposed shear flow with the shear rate $\dot{\gamma}$ and the shear plane perpendicular to the z-direction. $\hat{\boldsymbol{e}}_x$ denotes the unit vector along the x-axis. Finally, $G_{ij}(\boldsymbol{y},\boldsymbol{x}) \coloneqq \delta_{ij}/r + r_i r_j / r^3$ is the free-space Green's function, whereas $\boldsymbol{r} \coloneqq \boldsymbol{y} - \boldsymbol{x}$ and $r \coloneqq |\boldsymbol{r}|$. Thus, given the traction jump $\triangle \boldsymbol{f}^{\mathrm{h}}$ via formula (4), this integral equation allows us to compute the velocity at each node of the capsule's surface. Afterwards, they are moved with the flow according to the kinematic (no-slip) condition [44]

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}) \,, \quad \boldsymbol{x} \in S \,, \tag{23}$$

where t denotes the time.

Methods A – E. The first five Methods A – E directly use the discretization of the surface S with flat triangles. Integrals are performed by a Gaussian quadrature with 7 points per triangle [117]. Necessary quantities at these points are obtained via linear interpolation from the nodes [43]. The polar integration rule is used for singular triangles [118]. The ordinary differential equation (23) is solved using the Cash-Karp method [119] which is an explicit embedded Runge-Kutta scheme of order four and five, i.e. an adaptive step size algorithm. Its relative tolerance is fixed to 10^{-7} while the absolute tolerance is $10^{-6}R$. Decreasing both by a factor of 10 did not change the results significantly. Note that the mesh remains very homogeneous throughout the whole simulation thanks to the elastic forces acting between the nodes; no additional mesh control scheme was necessary.

Solving equation (22) can lead to a volume drift (a mere discretization artifact). To counter it, we first rotate the $3N$-dimensional solution vector onto the hyperplane defined by the discretized version of the no-flux condition $\oint_S \boldsymbol{u} \cdot \boldsymbol{n}\,\mathrm{d}S = 0$, similar to the approach employed in reference [68]. This reduces the drift but cannot eliminate it completely. Hence, after each time step we additionally rescale the object as described in [68]. This leads to a perfect conservation of volume.

Method S. Method S uses a completely different code basis and is entirely based on Loop's subdivision surface algorithm (cf. section 2.5). I.e. equation (22) is computed using the smooth limit surface. Triangles, where the Green's function has a singularity, are treated as in reference [68]. The time evolution (23) is solved using the trapezoidal rule, a fully implicit scheme. The time step size is fixed to $\dot{\gamma}\triangle t = 10^{-4}$. No remeshing was performed during the simulations. Furthermore, the volume drift remained below 0.005 % (320 triangles) and 0.0002 % ($\geqslant$ 1280 triangles) during all simulations. See reference [70] for more details.

4.3.2. Lattice-Boltzmann

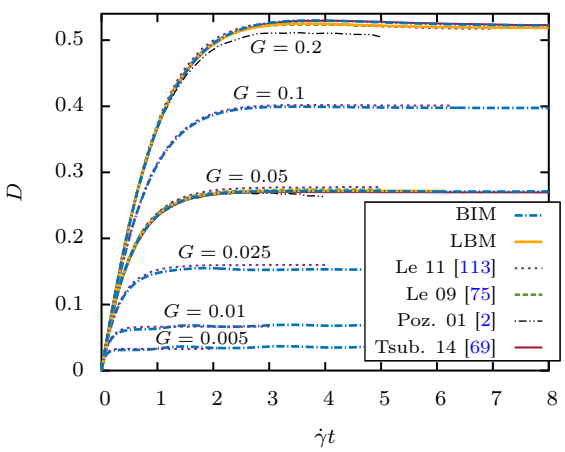
The Lattice-Boltzmann method is a mesoscopic method for solving fluid problems which is based on a discretization of space and velocities. Over the last couple of years it has become a well-established method. We omit the details here. They can be found for example in references [45–47]. We use the D3Q19 scheme as provided by the ESPResSo package [120, 121].

For the capsule we implemented the Immersed Boundary Method (IBM) into ESPReSo [120], following mostly the works of Krüger [28, 48]. Besides the elastic and bending forces, we add an additional force to minimize deviations from the initial volume as in [28], controlled by the modulus κ_V. The source code is publicly available in the current development branch of ESPReSo.

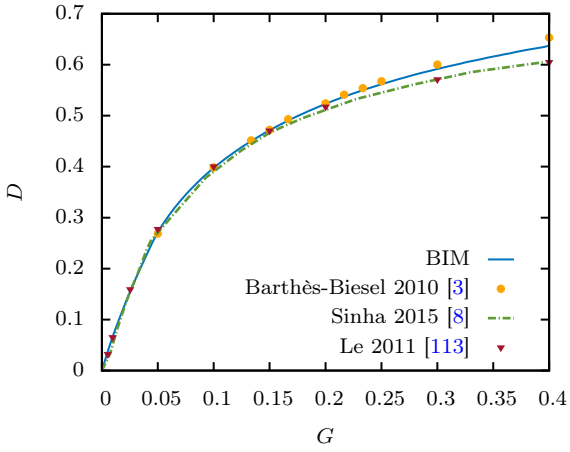
To solve the dynamics, the explicit Euler scheme with the time step set to $\dot{\gamma}\triangle t = 2.5\times10^{-5}$ and the LBM grid size to $13.5/R \equiv 1$ are used. Contrary to BIM, periodic boundary conditions are (necessarily) employed. The simulation box size is set to $9.5R \times 9.5R$ in the lateral direction and $19.0R$ in height (changing it does not alter the results significantly). The shear flow is realized by placing two plane parallel walls moving in opposite directions at the top and bottom of the simulation box, implemented with the bounce-back boundary condition. Before inserting the sphere in the center of the box, we wait until the shear flow is fully developed. The Reynolds number $\mathrm{Re} = Rv/\nu$, with v the velocity of the walls and ν the kinematic viscosity, is always smaller than 0.3. The lattice Mach number is always smaller than 0.01. Finally, the modulus of the volume conservation force is fixed to $\kappa_\mathrm{V} = 100\kappa_\mathrm{S}$ (in simulation units), leading to a maximal volume drift of less than 0.1 % in the presented simulations.

4.4. Verification of the codes

Extensive tests were carried out to ensure the correctness of our three simulation codes. An example for $\hat{\kappa}_\mathrm{B} = 0$ (i.e. without any bending) can be found in figure 12 that compares the BIM code of Methods A – E with several references found in the literature. Some of them use vastly different simulation methodologies. Notice that the simulations remain stable despite the occurrence of buckling thanks to the small effective bending rigidity inherent to finite element methods [3, 55, 76]. A comparison with the LBM code can be found in figure 12a. For $\hat{\kappa}_\mathrm{B} \neq 0$ consider figure 17. Method S can be seen to produce identical results in figure 14.



(a) Time evolution of the Taylor deformation parameter D for various dimensionless shear rates G. Results by Huang *et al.* 2012 [71], Le 2010 [4] and Zhu 2015 [10] are not included for clarity but match well with our BIM.



(b) Stationary value of the deformation parameter as a function of the shear rate. The stationary value is computed by averaging the deformation parameter for $\dot{\gamma}t \in [5; 10]$.

Figure 12: Time evolution and stationary values of the Taylor deformation parameter D for a spherical capsule without bending resistance ($\hat{\kappa}_B = 0$) and 5120 triangles in shear flow. We checked that the inhomogeneous mesh with 3914 triangles leads to similar results for $G = 0.05$ and $G = 0.2$. Comparison with the values of Le *et al.* 2011 [113] (projection method & IBM, subdivision surface), Le *et al.* 2009 [75] (projection method & IBM, curved triangles), Pozrikidis 2001 [2] (BIM, curved triangles), Tsubota 2014 [69] (BIM, flat triangles), Barthès-Biesel *et al.* 2010 [3] (BIM, B-Splines) and Sinha *et al.* 2015 [8] (BIM, flat triangles).

All in all, the images show very good agreement, both between BIM and LBM as well as with the literature. The results were checked to be well converged, as also shown in the next section. We remark that the data in section 3 was produced using the code basis of the two BIM implementations.

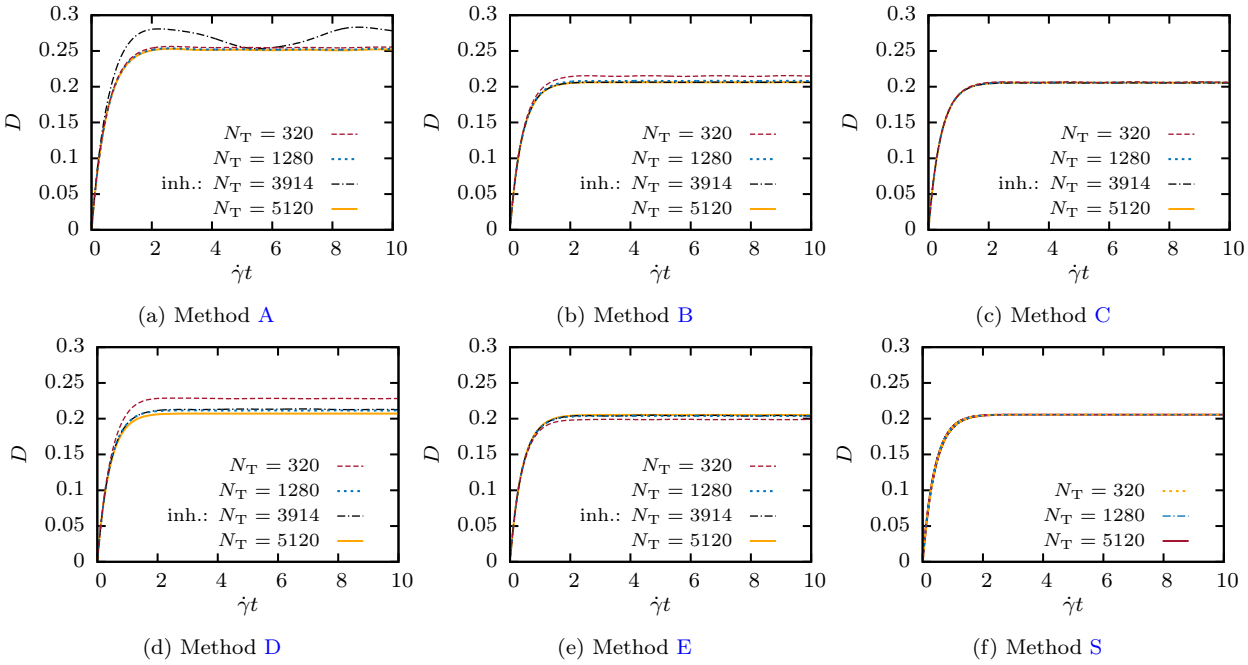
4.5. Analysis of the different bending algorithms

4.5.1. Convergence with resolution

We now compare the performance of the different bending algorithms for the capsule with both shear and bending resistance. As a starting point we investigate the convergence of each method with respect to the number of triangles used to discretize the surface. Choosing $G = 0.2$ and $\hat{\kappa}_B = 0.15$, the results in figure 13 are obtained.

Method A appears to be rather insensitive to the resolution which is somewhat surprising considering the increase of its error with the number of triangles (figure 10 from page 15). However, it has serious troubles with the inhomogeneous mesh, just like in the analytic comparison, leading to an oscillatory graph. In 3D this is notable as a surface with slight "bumps". On the other hand, Method C and especially S are the fastest converging algorithms, giving similar results as the other approaches but already at 320 triangles. This observation fits well with the average error in figure 10b which roughly remained constant. Furthermore, this also indicates that indeed the average rather than the maximum error is the more meaningful measure for the present setup. After all, the maximum error diverged as seen in figure 10a, but this behavior is not reflected here. However, C obtained very high errors for the inhomogeneous triangulation even in the average measure, although in shear flow no effect is observed (figure 13c). Method B is slightly inferior regarding convergence compared to C, but otherwise identical. E reaches its limit at $N_T = 1280$. The slowest convergence is exhibited by Method D.

We finally note that very similar observations are made for $G = 0.05$ and $\hat{\kappa}_B = 0.0375$ and for the inclination angles as depicted in the supplementary information. The same holds for meshes of type MT2.



(a) Method A (b) Method B (c) Method C

(d) Method D (e) Method E (f) Method S

Figure 13: Taylor deformation parameter as a function of time after turning on the shear flow for BIM. $G = 0.2$ and $\hat{\kappa}_B = 0.15$. We find that for almost all algorithms 320 triangles are too coarse to obtain converged results, whereas the curves for 1280 and 5120 triangles are almost indistinguishable. Method A shows significant changes for the inhomogeneous mesh with 3914 triangles. Graphs of the inclination angle and results for a different shear rate can be found in the supplementary material.

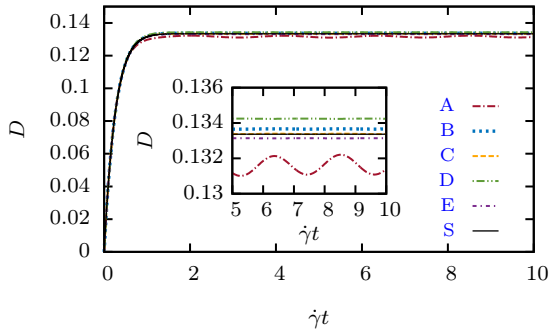
4.5.2. Direct comparison of the methods

A direct comparison of the six algorithms for a resolution of 5120 triangles using BIM is shown in figure 14. For the small shear rate and bending modulus in figure 14a only minor differences are observed. This shows that for small deformations the actual method plays only a secondary role, at least for sufficiently homogeneous meshes. However, larger deviations are seen if the parameters are increased to $G = 0.2$ and $\hat{\kappa}_B = 0.15$ as shown in figure 14b. Most notably, Method A deviates strongly from the other five methods. This might be understandable from the large errors that were observed in section 3 for this algorithm. Tsubota [69] also noticed major differences between Methods A and C. Furthermore, B – E and S coincide almost perfectly, with deviations in the $< 1\,\%$ regime. For these we observed average errors that differ by approximately one order of magnitude (figure 10b on page 15), so this is somewhat surprising. We conclude that all methods except A are roughly equally well suited for this setup.

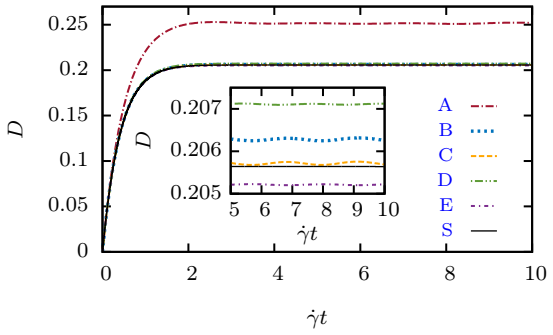
To assess a possible influence of the flow solver, we also simulate the system with Lattice-Boltzmann, restricting ourselves here to Methods A and B which directly yield the nodal forces $\boldsymbol{F}$ as required by our LBM implementation. The results in figure 15 show very good agreement with the BIM data, proofing that the observed differences are inherent to the bending algorithms themselves and largely independent of the flow solver.

4.5.3. Performance considerations

An important criterion in the selection of the most suitable bending algorithm for a given problem will be its execution speed. The approaches examined in the present work exhibit significantly different performance characteristics. Most notably the computational complexity of Method D and S are $\mathcal{O}(N^2)$ while the remaining algorithms are $\mathcal{O}(N)$. Indeed, all methods except D and S require (mostly) only the first ring of neighbors. We refrain here from comparing absolute execution times as varying degrees of optimization levels (caching of quantities, parallelization, SIMD vectorization, etc.) and the hardware may strongly influence these times. But in most cases one expects to find that *one* evaluation of Method D or S

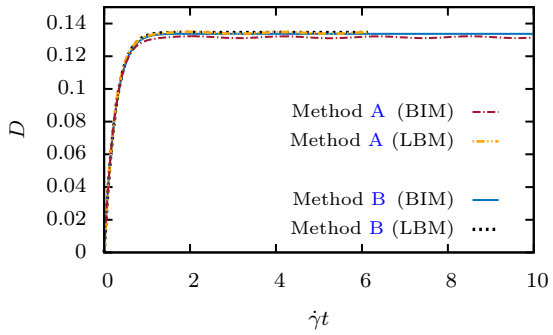


(a) $G = 0.05$, $\hat{\kappa}_B = 0.0375$

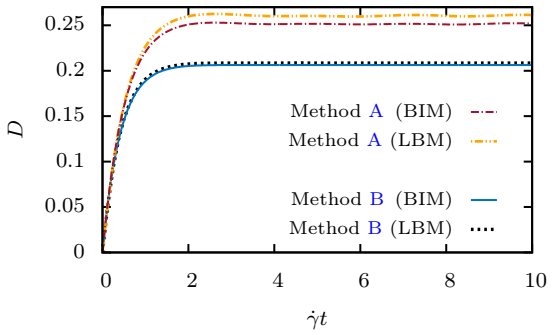


(b) $G = 0.2$, $\hat{\kappa}_B = 0.15$

Figure 14: Direct comparison of all bending algorithms for BIM and 5120 triangles. Insets: Magnification of $\dot{\gamma}t \in [5; 10]$.



(a) $G = 0.05$, $\hat{\kappa}_B = 0.0375$



(b) $G = 0.2$, $\hat{\kappa}_B = 0.15$

Figure 15: Comparison of the LBM results with BIM for two different shear rates and bending moduli. In both cases 5120 triangles were used.

Triangles	$\hat{\kappa}_\mathrm{B} = 0$	A	B	C	D	E
320	0.081	0.016	0.060	0.057	0.085	0.057
1280	0.064	0.0015	0.0057	0.0056	0.063	0.0060
5120	0.029	0.00014	0.00053	0.00052	0.0085	0.00056
3914 (inh.)	0.023	0.00014	0.00014	0.00040	0.013	0.0017

(a) Average $\dot{\gamma}\triangle t$ for $G = 0.05$, $\hat{\kappa}_\mathrm{B} = 0.0375$.

Triangles	$\hat{\kappa}_\mathrm{B} = 0$	A	B	C	D	E
320	0.20	0.013	0.052	0.050	0.15	0.048
1280	0.16	0.0012	0.0050	0.0049	0.078	0.0051
5120	0.13	0.00011	0.00046	0.00046	0.0076	0.00048
3914 (inh.)	0.11	0.00012	0.000023	0.00035	0.011	0.0013

(b) Average $\dot{\gamma}\triangle t$ for $G = 0.2$, $\hat{\kappa}_\mathrm{B} = 0.15$.

Table 4: Average dimensionless time step size $\dot{\gamma}\triangle t$ for the different bending algorithms and resolutions as chosen by the Cash-Karp algorithm for BIM. The relative tolerance of the time stepping scheme is 10^{-7} and the absolute tolerance is $10^{-6}R$. We also included the data without any bending ($\hat{\kappa}_\mathrm{B} = 0$ column). The corresponding deformations can be seen in figures 12a, 13 and the supplementary information. The average is performed over $\triangle t$ in the range $\dot{\gamma}t \in [5; 10]$.

is the slowest (due to the inferior scaling), followed by Method E (because the fitting procedure involves solving several small linear systems), and the remaining algorithms being the fastest.

Instead, we employ an implementation-independent indicator, namely the required time step $\triangle t$ to remain in the stable region. In general, stiffer systems require smaller time steps if explicit integrators are used. Since we use an explicit adaptive time stepping algorithm for the BIM implementation of Methods A – E, the automatically chosen values for $\triangle t$ can serve as an indicator for the stiffness and the overall performance of dynamic simulations. Table 4 shows the corresponding data for two different shear rates and bending moduli.

We first note that an increase of triangle count by a factor of four leads to a decrease of the step size by roughly one order of magnitude for all algorithms – except for Method D. This algorithm also exhibits the largest of all time steps. This can be attributed to its already mentioned robustness which in turn originates from its large support (cf. sections 2.4.2 and 3.5). The smallest time step is required by Method A, being roughly a factor of 3 smaller than for B, C and E (which all have approximately the same $\triangle t$). Moreover, the inhomogeneous mesh takes its toll for Methods A, C and especially B, leading to step sizes even below or equal to the one for 5120 triangles. Out of the three, C performs the best but cannot compete with E or even D.

The general effect of decreasing step sizes with resolution is explained by Boedec *et al.* [58]: More triangles imply that oscillatory modes with shorter wavelengths are resolved, but shorter wavelengths in turn mean faster typical time scales for the bending forces. Hence, the more triangles, the smaller the shortest occurring time scales and thus the smaller the necessary $\triangle t$.

Furthermore, considering the data without any bending ($\hat{\kappa}_\mathrm{B} = 0$), one notices the tremendous effect the inclusion of bending effects has on the step size and thus on the overall performance. For mediocre resolutions, $\triangle t$ is often two orders of magnitude larger than with bending. This relates to the general stiffness of the appearing fourth derivative. A usual remedy is to work with an implicit or semi-implicit time stepping scheme, as we did in Method S. See for example references [58, 77].

4.5.4. Comparison with the literature

Before we can compare our simulations with the literature, we comment on the constitutive bending law. Namely, some references use the linear relation [2, 10, 20, 75]

$$\boldsymbol{m} = \kappa_\mathrm{B}\boldsymbol{B} \tag{24}$$

for the bending moment $\boldsymbol{m}$, where $\boldsymbol{B}$ is the Cartesian curvature tensor. Pozrikidis [2] showed that this model is equivalent to the Helfrich law from equation (1) (also see [53]). Similar, he remarks that for zero reference curvatures the model introduced by Zarda *et al.* [122] is the same, too.

Keeping this in mind, we compare the results from the literature in figures 16 and 17. The data obtained using Method C (BIM with flat triangles) and Method S (BIM with subdivision surfaces) is also included. Both are representative for the remaining algorithms as well as the LBM flow solver as has been shown in figure 14, with the sole exception of Method A.

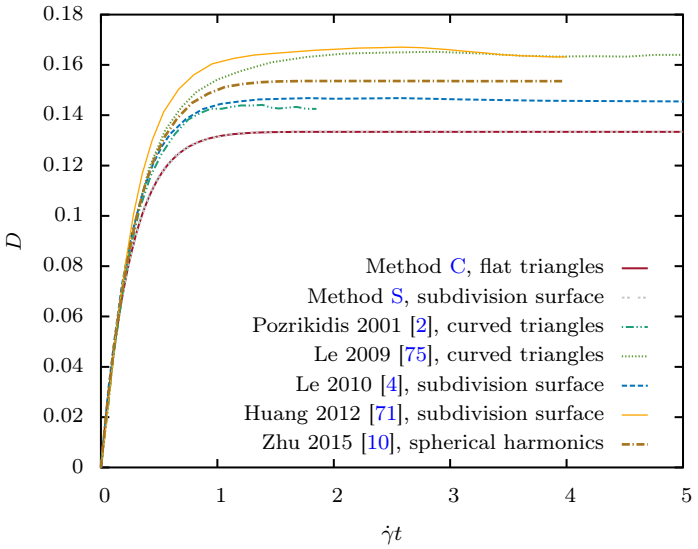


Figure 16: Comparison of the deformation parameter for $G = 0.05$ and $\hat{\kappa}_\mathrm{B} = 0.0375$ from the recent literature with Method C (BIM, 5120 flat triangles) and Method S (BIM, subdivision surface with 5120 elements). All references use the same physical parameters, except for Pozrikidis [2] who used a Hookean elastic law. All use the Helfrich or the equivalent linear bending model from equation (24), except Huang *et al.* [71] who employs the also equivalent bending model introduced by Zarda *et al.* [122] (see main text). However, the employed flow and discretization algorithms vary greatly. We note that our LBM simulations give the same results as Method C (fig. 15) and that both BIM and LBM agree well without bending with the literature as shown in section 4.4.

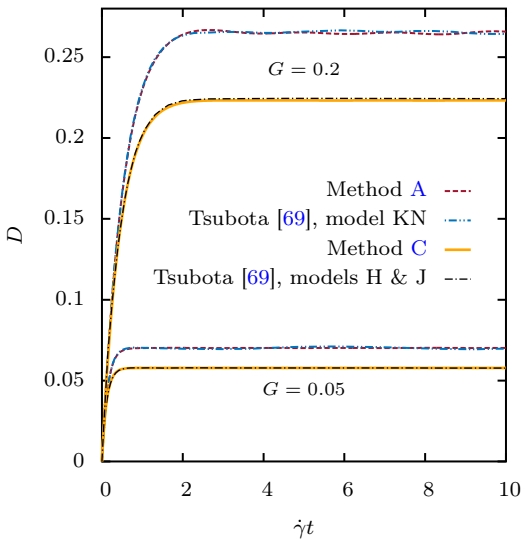


Figure 17: Comparison of the deformation parameter for $\hat{\kappa}_\mathrm{B} = 2/15$ (once for $G = 0.05$ and once for $G = 0.2$) for Methods A and C with Tsubota 2014 [69]. 5120 triangles. Method A is virtually identical to model KN from [69], Method C is essentially alike to model H. Note that Tsubota uses BIM, 4604 flat triangles and the same physical conditions.

Observing figure 16, we note that the values scatter a lot ($\lesssim 20\,\%$). The source is not easily identified. The algorithm for the shear elasticity and the flow solver should have no influence, as we observe good agreement between our BIM and LBM (fig. 15) and with the literature for $\hat{\kappa}_\mathrm{B} = 0$ (fig. 12) for all references. We also rule out errors in our implementation of the bending for Methods A and C: Tsubota [69] employs virtually the same algorithms and discretizations as we do, and his results agree with ours extremely well (figure 17). Also Method S, which uses a completely different code basis and, additionally, subdivision surfaces, matches almost perfectly with the remaining methods except A. Furthermore, we carefully checked that (apart from the explicitly mentioned differences) all references use the same physical laws.

Considering the agreement for $\hat{\kappa}_\mathrm{B} = 0$, this once again emphasizes the huge difficulty inherent in the computation of the bending forces.

5. Conclusion

To summarize, we presented six different algorithms to compute the bending forces on 3D meshes discretized with flat triangles. They are all based on the famous Canham-Helfrich constitutive law for the

bending energy, but differ in their numerical implementation, using well-known ingredients and new developments. The methods, denoted by A – E and S, can be sorted into three different categories, depending on the variational derivative being performed before ("force formulation") or after ("variational formulation") the surface discretization. Method S is somewhat set apart from the others because it uses the finite element method to obtain the force density. Their characteristics were collected in table 1 (page 4). In short: Method A contains a sum over all angles between the triangles' normal vectors, whereas Method B uses a variant of the cotangent scheme. The same holds for Method C, except that it employs the variational formulation and a slightly different measure for the area per node. Moreover, Method D is based on a kernel of the heat equation, Method E fits multiple parabolas onto the surface components, and Method S uses a subdivision scheme.

We then analyzed the behavior of the algorithms quantitatively by comparing their various components such as the mean curvature H or the Laplace-Beltrami operator of H (essentially a fourth order derivative) and finally the bending force itself with analytic results obtained for the typical red blood cell shape. A concise overview of the results was given in tables 2 and 3 (page 17). Regarding the maximum error, only Method D provides an acceptable approximation, being also the most robust on inhomogeneous meshes. No method converged at all nodes. If the errors are averaged over the entire mesh, Methods B – E and S give acceptable errors, but only D actually shows convergence. In general we found that the more vertices an algorithm takes into account to compute the values at a single node, the better the overall robustness. Hence, all Methods except D depict high sensitivity to the regularity of the mesh. No major quality differences regarding the underlying principle of force and variational formulations (before and after discretization) were observed.

As a physical application we considered an elastic capsule in a viscous shear flow. All approaches gave similar results with the exception of Method A which showed significant deviations. Still this illustrates that the behavior of some single individual points is of only minor concern for the purpose of hydrodynamics in the small Reynolds number regime. Furthermore, a small review of the existing literature employing different surface discretizations revealed large deviations for the hydrodynamic results, illustrating again the tremendous difficulty inherent with computing a fourth order derivative even on higher order surface approximations.

The results for the individual methods can be summarized as follows:

- Method A showed the largest errors of all the methods for the force density, quickly diverging in both the maximal and sometimes even the average error measure. The hydrodynamics were noticeably different compared to the other algorithms, especially for the inhomogeneous mesh. Hence it is very sensitive to irregularities. Furthermore, it often required the smallest step size in order to remain in the stable region. Even the theoretical relationship to Helfrich's law is somewhat "blurred" because of the shape-dependent relationship between the numerical parameter $\widetilde{\kappa}_\mathrm{B}$ from equation (6) and the physical bending modulus κ_B. On the other hand, it is the most easily implemented method.

- Method B turned out to be similar to Method C for homogeneous meshes, but is somewhat worse for inhomogeneous triangulations regarding errors and required step size.

- Method C depicted diverging behavior for the maximal error measure of the force density $\boldsymbol{f}$, while the deviations stayed roughly constant in the average measure. However, the algorithm displayed troubles with the inhomogeneous mesh. The hydrodynamical results were very similar to Methods B – E and S, and the required step size was comparable to B and E.

- Method D was by far the most robust, showing convergence for $\boldsymbol{f}$ on the regular meshes (albeit with a systematic deviation in the maximal error measure) and working reasonably well on the inhomogeneous one. It also leads to the largest step sizes. Unfortunately, one evaluation is very expensive since it scales as $\mathcal{O}(N^2)$ where N are the number of nodes.

- Method E is similar to C as the maximal errors for $\boldsymbol{f}$ diverged and the average remained roughly constant. However, it handled the irregular mesh better than Methods A – C but still worse than D. The required step size was comparable to Methods B and C.

- Method S in general showed behavior alike to Method C. However, it provided errors that were significantly smaller. The complexity is $\mathcal{O}(N^2)$.

When taking the results for the systems investigated here as general guidelines, we make the following recommendations if one wishes to implement the Helfrich law for bending forces on triangulated meshes:

- Method D is in principle the best algorithm due to its robustness and often acceptable convergence properties. It should be chosen if performance is no issue and the resolution is sufficiently high. Therefore it is currently the best choice for computer graphics applications. Unfortunately, it is most likely too slow for dynamic simulations, which also often need to work with relatively coarse meshes where it performed below average.

- Method S is a good choice for homogeneous meshes due to the comparably small errors, although it does not provide proper convergence. It should especially be chosen over E if other parts of the numerical algorithm (such as surface integrals) can benefit from the subdivision surface representation. The $\mathcal{O}(N^2)$ scaling has a small prefactor, which means that it is no bottleneck for practical resolutions. The major error source were nodes without six neighbors, indicating that it is not well suited for more irregular meshes.

- Method E is the second best choice if one wishes to stay with flat triangles in all parts of the numeric implementation. The reason is that it is also relatively robust, but has worse convergence properties than D (similar to S). Compared to e.g. algorithm C, the errors for $\boldsymbol{f}$ were always smaller. Furthermore, it has a much better computational complexity than Method D, making it suitable for larger simulations.

- Method C is inferior to Method E and S regarding absolute errors, and hence E or S are usually preferable. The exception might be if performance issues arise: C can be implemented more efficiently since it does not involve solving linear systems. We remark that it provides very good results for the shear flow setup, even for more irregular triangulations.

- Method B has similar stiffness and implementation characteristics as Method C, but performs worse especially for the inhomogeneous mesh. Depending on the application, the only advantage compared to C might be that it directly yields a force rather than a force density.

- Method A can be used if the goal is not to approximate the Helfrich model but to just include "some" bending resistance, i.e. in case Method A is taken as the *model itself*. Another useful application would be to prevent the mesh from buckling e.g. in shear flow simulations (i.e. for the purpose of numerical stability [29]). In this case we suggest to use the small angle approximation from section 2.3.1 for efficiency reasons. Note that the bending modulus must be chosen sufficiently small in order to keep the required step size large and the physical impact as small as possible. In any other case, one of the other algorithms should be preferred.

As a future research direction, a promising approach would be to mix the algorithms. For example, Method E often showed the smallest errors for the mean curvature H, while Method D is the most robust. Hence it might be worthwhile to use approach E for H and D for the Laplace-Beltrami operator of H.

Acknowledgments

We would like to thank G. Boedec, A. Farutin and M. Jaeger for helpful discussions. Funding from the Volkswagen Foundation and computing time granted by the Leibniz-Rechenzentrum on SuperMUC is gratefully acknowledged by A. Guckenberger, M. Schraml and S. Gekle. Funding and support from the KONWIHR network is appreciated by A. Guckenberger and S. Gekle. A. Guckenberger thanks the Elitenetzwerk Bayern (ENB), Macromolecular Science, for their support. P. G. Chen and M. Leonetti have benefited from financial support from the ANR Polytransflow (Grant No. 13-BS09-0015-01), from Labex

MEC (Grant No. ANR-11-LABX-0092), from A*MIDEX (Grant No. ANR-11-IDEX-0001-02), and from CNES. This work was granted access to the HPC resources of Aix-Marseille Université financed by the project Equip@Meso (ANR-10-EQPX-29-01).

Appendix A. Derivation of the force for Method B

In this section we provide the derivation of the force $\boldsymbol{F}(\boldsymbol{x}^{(l)})$ at some node $\boldsymbol{x}^{(l)}$, $l = 1, \ldots, N$ via Method B from section 2.3.2. N is the number of surface nodes. We start by writing equation (8) as

$$E_{\mathrm{B}} \approx \frac{\kappa_{\mathrm{B}}}{2} \sum_{i=1}^{N} \widetilde{E}_{\mathrm{B}}^{(i)} \,, \tag{A.1}$$

where

$$\widetilde{E}_{\mathrm{B}}^{(i)} \coloneqq 2 \frac{\left[\sum_{j(i)} \left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right) T_{ij}\right]^2}{\sum_{j(i)} l_{ij}^2 T_{ij}} \,, \quad i = 1, \ldots, N \,, \tag{A.2}$$

the sums with $j(i)$ are sums over all neighbors j of node i, and

$$T_{ij} \coloneqq \cot \vartheta_1^{(ij)} + \cot \vartheta_2^{(ij)} \,, \qquad\qquad l_{ij} \coloneqq |\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}| \,. \tag{A.3}$$

The angles $\vartheta_1^{(ij)}$ and $\vartheta_2^{(ij)}$ were already defined in section 2.3.2. See figure 1 (page 5) for a sketch. We now require an analytic expression for the force $\boldsymbol{F}(\boldsymbol{x}^{(l)})$ from equation (5) at each node $\boldsymbol{x}^{(l)}$, with the energy from equation (A.1). For this, we compute the k'th component of the gradient of $\widetilde{E}_{\mathrm{B}}^{(i)}$ with respect to the vertex $\boldsymbol{x}^{(l)}$:

$$\begin{aligned} \frac{\partial \widetilde{E}_{\mathrm{B}}^{(i)}}{\partial x_k^{(l)}} = {} & 4 \frac{\sum_{j(i)} \left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right) T_{ij}}{\sum_{j(i)} l_{ij}^2 T_{ij}} \cdot \sum_{j(i)} \left[\left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right) \frac{\partial T_{ij}}{\partial x_k^{(l)}} + T_{ij} \hat{\boldsymbol{e}}_k \left(\delta_{il} - \delta_{jl}\right) \right] \\ & - 2 \frac{\left[\sum_{j(i)} \left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right) T_{ij}\right]^2}{\left(\sum_{j(i)} l_{ij}^2 T_{ij}\right)^2} \sum_{j(i)} \left[T_{ij} \frac{\partial (l_{ij}^2)}{\partial x_k^{(l)}} + l_{ij}^2 \frac{\partial T_{ij}}{\partial x_k^{(l)}} \right] \,. \end{aligned} \tag{A.4}$$

Here, $\hat{\boldsymbol{e}}_k$ is the k'th canonical unit vector and δ_{il} the Kronecker symbol. Continuing, we find

$$\frac{\partial (l_{ij}^2)}{\partial \boldsymbol{x}^{(l)}} = 2(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)})(\delta_{il} - \delta_{jl}) \,. \tag{A.5}$$

Next, we need to express T_{ij} and therefore the angles $\vartheta_{1,2}^{(ij)}$ through the nodes' positions. Thus, we define

$$\chi_{i,j,j-1} \coloneqq \cos \vartheta_1^{(ij)} = \frac{(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j-1)}) \cdot (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(j-1)})}{l_{i,j-1} l_{j,j-1}} \quad \text{and} \tag{A.6a}$$

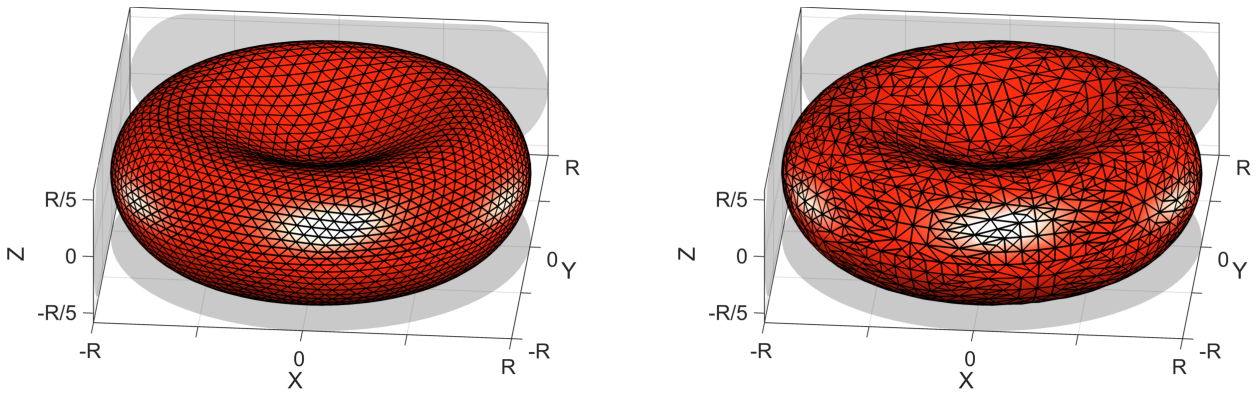
$$\chi_{i,j,j+1} \coloneqq \cos \vartheta_2^{(ij)} = \frac{(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j+1)}) \cdot (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(j+1)})}{l_{i,j+1} l_{j,j+1}} \,, \tag{A.6b}$$

where $j-1$ specifies the "previous" and $j+1$ the "next" node relative to node j of the ring-1 neighbors of vertex i, as seen in figure 1. Circular enumeration is implied. We can now exploit $\vartheta_{1,2}^{(ij)} \in \,]0, \pi[$ and write

$$T_{ij} = \frac{\chi_{i,j,j-1}}{\sqrt{1 - \chi_{i,j,j-1}^2}} + \frac{\chi_{i,j,j+1}}{\sqrt{1 - \chi_{i,j,j+1}^2}} \,. \tag{A.7}$$

This leads to

$$\frac{\partial T_{ij}}{\partial \boldsymbol{x}^{(l)}} = \frac{1}{\left(1 - \chi_{i,j,j-1}^2\right)^{3/2}} \frac{\partial \chi_{i,j,j-1}}{\partial \boldsymbol{x}^{(l)}} + \frac{1}{\left(1 - \chi_{i,j,j+1}^2\right)^{3/2}} \frac{\partial \chi_{i,j,j+1}}{\partial \boldsymbol{x}^{(l)}} \,, \tag{A.8}$$



(a) Homogeneous mesh MT2 with 5120 triangles, obtained by refining an icosahedron by placing new nodes in the middle of edges and moving them out onto the sphere.

(b) Inhomogeneous mesh with 3848 triangles derived from MT2, obtained via Rivara's longest-edge bisection algorithm.

Figure B.18: Illustrations of the typical RBC shape discretized with the MT2 approach. See figure 2 for the MT1 meshes.

whereas for $m = j - 1$ or $m = j + 1$ we find

$$\frac{\partial \chi_{i,j,m}}{\partial \boldsymbol{x}^{(l)}} = \frac{1}{l_{im} l_{jm}} \Big[(\delta_{il} - \delta_{ml}) (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(m)}) + (\delta_{jl} - \delta_{ml}) (\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(m)}) - \\ - \frac{l_{jm}}{l_{im}} \chi_{ijm} (\delta_{il} - \delta_{ml}) (\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(m)}) - \frac{l_{im}}{l_{jm}} \chi_{ijm} (\delta_{jl} - \delta_{ml}) (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(m)}) \Big] . \tag{A.9}$$

Substituting equation (A.9) into (A.8), and then (A.8) and (A.5) into (A.4) gives the contribution of node $\boldsymbol{x}^{(i)}$ (and its neighbors) to the force acting on node $\boldsymbol{x}^{(l)}$. The total force $\boldsymbol{F}(\boldsymbol{x}^{(l)})$ then follows from summing over all these contributions and multiplying the result with $-\frac{\kappa_B}{2}$. Obtaining the force density $\boldsymbol{f}$ from $\boldsymbol{F}$ was explained in section 2.6.

Appendix B. MT2 mesh

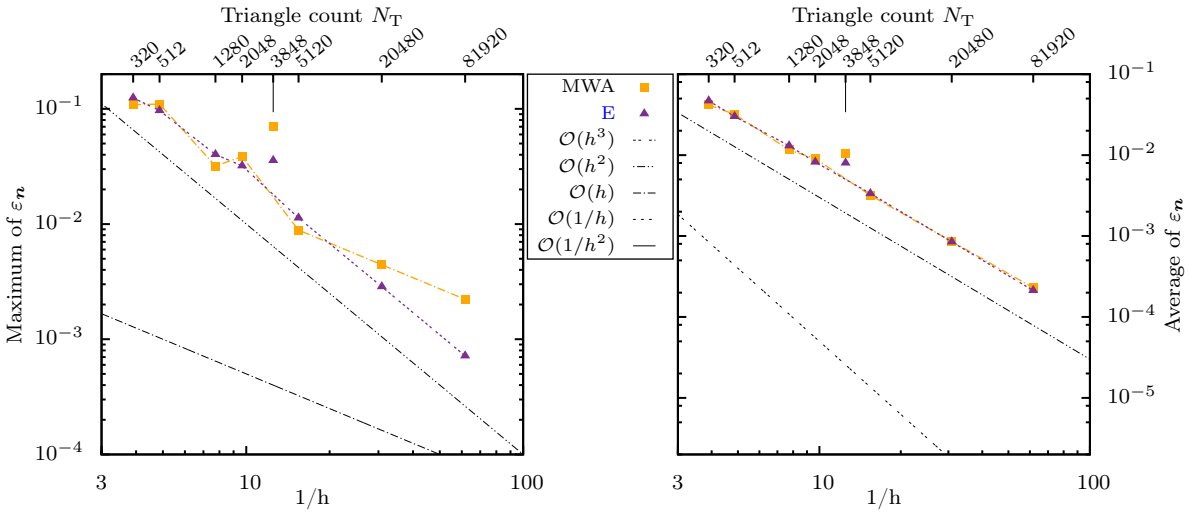
Section 3 from the main text presented the maximal and average errors for the various Methods for the typical RBC shape using MT1 meshes. Here we provide the same figures as in section 3 for Methods A – E with the MT2 mesh. We highlight the major differences below.

As a start, figure B.18 shows 3D images of the homogeneous and inhomogeneous MT2 discretizations. Compared to the MT1 versions from figure 2, the triangles around nodes with only five neighbors are somewhat larger. Note that the inhomogeneous mesh (figure B.18b) has 3848 rather than 3914 triangles because the edge lengths are different. Hence, the splitting order of the edges in Rivara's algorithm is different, and therefore also the final mesh.

The errors for the normal vector are depicted in figure B.19. We remark that the MT2 mesh reduces the convergence rate of the maximal error for the MWA algorithm to $\mathcal{O}(h)$ at smaller resolutions than the MT1 mesh.

The results for the mean curvature in figure B.20 do not differ largely in the convergence rates compared to MT1, but in the absolute value of the maximal errors. Notably, Method C is almost up to one order of magnitude larger.

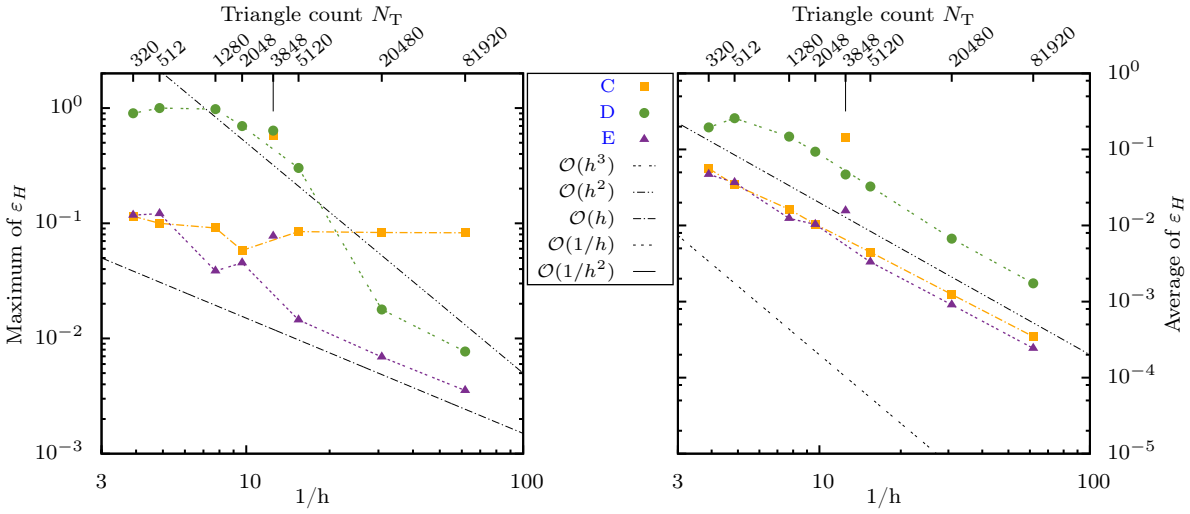
For the Laplace of the mean curvature, $\Delta_S H$, see figure B.21. Here, the maximal error for the otherwise very robust Method D starts to increase with resolutions beyond 20480 triangles (contrary to the MT1 version, where it stays constant). The rate for the average error is reduced from $\mathcal{O}(h^2)$ to $\mathcal{O}(h)$. Method E shows qualitatively different behavior for the average error, too: Rather than decreasing with $\approx \mathcal{O}(h^2)$, it remains constant beyond 5120 triangles, i.e. does not converge properly. Furthermore, the absolute values for Method C are often more than a factor of two larger.



(a) Maximum of all nodes.

(b) Average over all nodes.

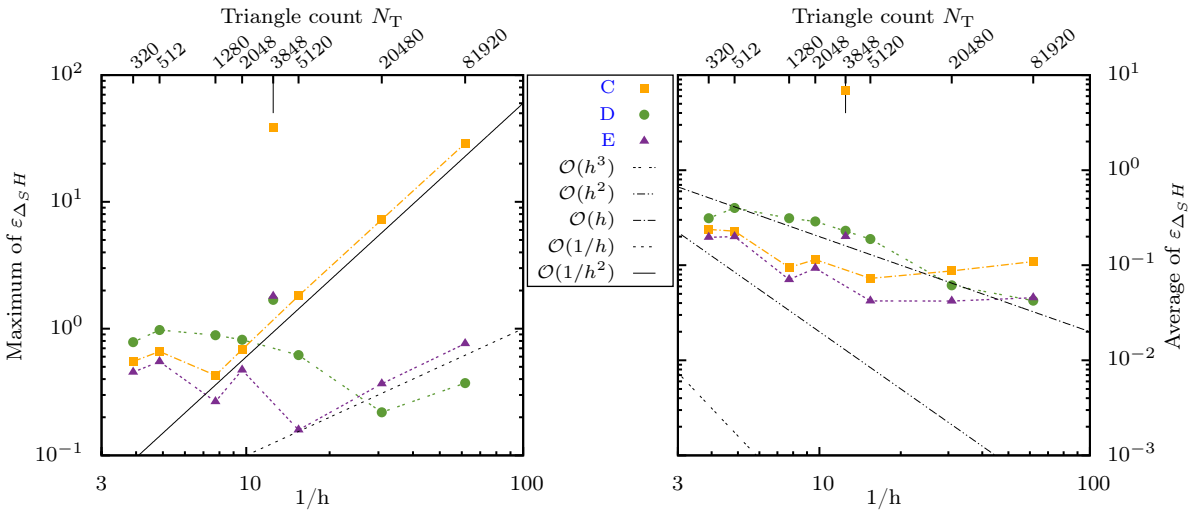
Figure B.19: The maximum and average of the relative error $\varepsilon_{\boldsymbol{n}}$ of the normal vector for the MT2 mesh. The version with the MT1 mesh can be found in figure 3, and the numerical values in the SI.



(a) Maximum of all nodes.

(b) Average over all nodes.

Figure B.20: The maximum and average of the relative error ε_H of the mean curvature for the MT2 mesh. The version with the MT1 mesh can be found in figure 5, and the numeric values in the SI.

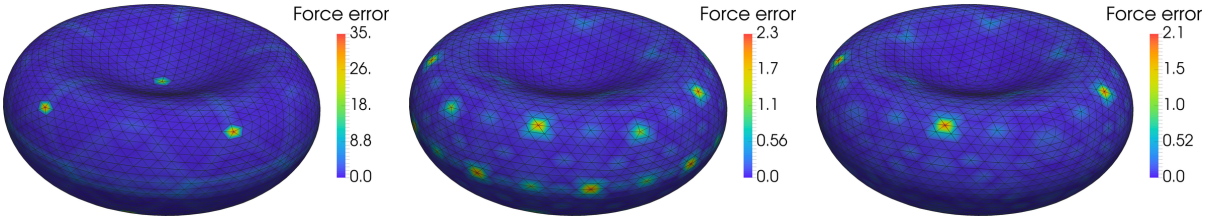


(a) Maximum of all nodes.

(b) Average over all nodes.

Figure B.21: The maximum and average of the relative error of the Laplace-Beltrami operator applied to H for the MT2 mesh. The version with the MT1 mesh can be found in figure 7, and the numeric values in the SI.

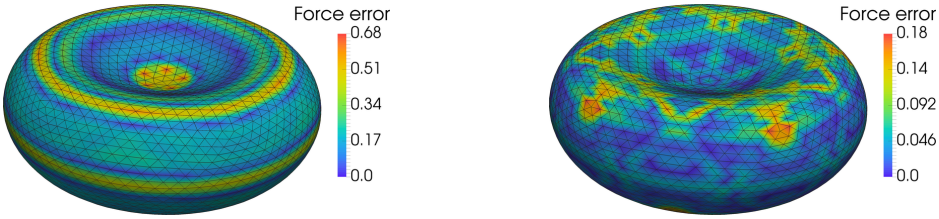
We finally consider the force density in figure B.23. Of course, the same observation as for $\Delta_S H$ hold for Methods C – E. Method B behaves the same as Method C. Moreover, Method A shows divergence with a rate of roughly $\mathcal{O}(h)$ for the average error, whereas in the MT1 version it stays approximately constant. Regarding the 3D patterns in figure B.22, the most notable change occurs for Method E. Rather than regular circles, more random patterns emerge. Inclusion of more rings, however, recovers circular patterns similar to Method D.



(a) Method A: As for MT1, nodes without six neighbors cause troubles.

(b) Method B: Compared to C additional problematic nodes appear.

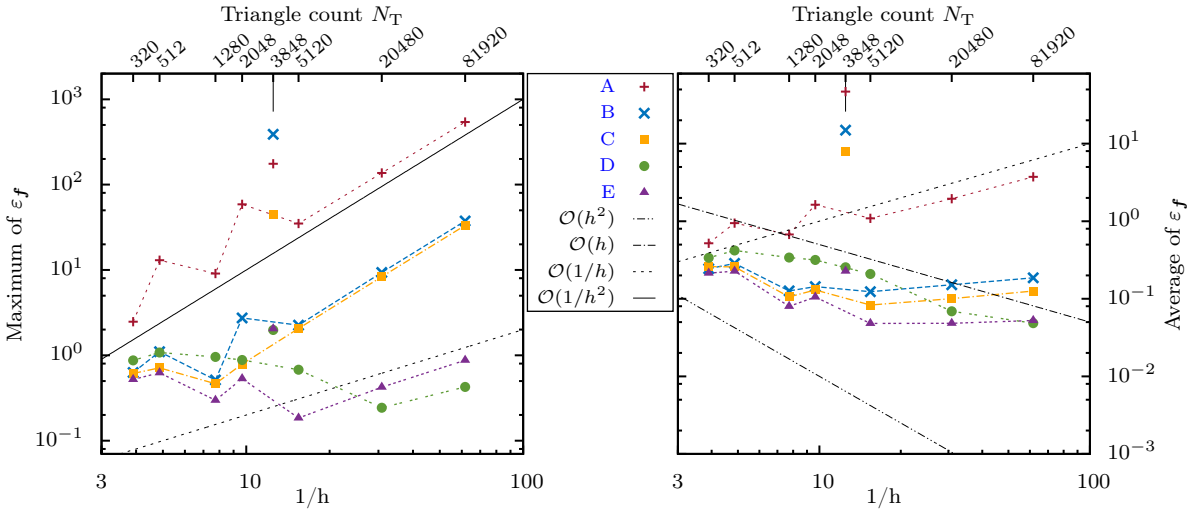
(c) Method C: The condition by Xu (cf. section 3.3) is violated most severely at the highlighted nodes.



(d) Method D: Very similar pattern as for MT1, illustrating its robustness.

(e) Method E: A notably different pattern when compared to MT1. But: Inclusion of more neighbor-rings leads again to rings (not shown).

Figure B.22: 3D illustration of the errors $\varepsilon_{\boldsymbol{f}}$ of the force density for 5120 triangles with the MT2 meshes. Note the scales of the color bars. The MT1 meshes were displayed in figure 9.



(a) Maximum of all nodes.

(b) Average over all nodes.

Figure B.23: The maximum and average of the relative error of the force density for the MT2 mesh. The version with the MT1 mesh can be found in figure 10, and the numerical values in the SI.

References

[1] M. Kraus, W. Wintz, U. Seifert, R. Lipowsky, Fluid Vesicles in Shear Flow, Phys. Rev. Lett. 77 (17) (1996) 3685–3688. doi:10.1103/PhysRevLett.77.3685.

[2] C. Pozrikidis, Effect of membrane bending stiffness on the deformation of capsules in simple shear flow, J. Fluid Mech. 440 (2001) 269–291. doi:10.1017/S0022112001004657.

[3] D. Barthès-Biesel, J. Walter, A.-V. Salsac, Flow-induced deformation of artificial capsules, in: C. Pozrikidis (Ed.), Computational Hydrodynamics of Capsules and Biological Cells, Chapman & Hall/CRC Mathematical and Computational Biology Series, CRC Press, Boca Raton, 2010, pp. 35–70. doi:10.1201/EBK1439820056-c2.

[4] D. V. Le, Effect of bending stiffness on the deformation of liquid capsules enclosed by thin shells in shear flow, Phys. Rev. E 82 (1) (2010) 016318. doi:10.1103/PhysRevE.82.016318.

[5] B. Kaoui, G. Biros, C. Misbah, Why Do Red Blood Cells Have Asymmetric Shapes Even in a Symmetric Flow?, Phys. Rev. Lett. 103 (18) (2009) 188101. doi:10.1103/PhysRevLett.103.188101.

[6] O. Aouane, M. Thiébaud, A. Benyoussef, C. Wagner, C. Misbah, Vesicle dynamics in a confined Poiseuille flow: From steady state to chaos, Phys. Rev. E 90 (3) (2014) 033011. doi:10.1103/PhysRevE.90.033011.

[7] C. Misbah, Vacillating Breathing and Tumbling of Vesicles under Shear Flow, Phys. Rev. Lett. 96 (2) (2006) 028104. doi:10.1103/PhysRevLett.96.028104.

[8] K. Sinha, M. D. Graham, Dynamics of a single red blood cell in simple shear flow, Phys. Rev. E 92 (4) (2015) 042710. doi:10.1103/PhysRevE.92.042710.

[9] D. A. Fedosov, M. Peltomäki, G. Gompper, Deformation and dynamics of red blood cells in flow through cylindrical microchannels, Soft Matter 10 (24) (2014) 4258–4267. doi:10.1039/C4SM00248B.

[10] L. Zhu, L. Brandt, The motion of a deforming capsule through a corner, J. Fluid Mech. 770 (2015) 374–397. doi:10.1017/jfm.2015.157.

[11] D. Abreu, M. Levant, V. Steinberg, U. Seifert, Fluid vesicles in flow, Adv. Colloid Interface Sci. 208 (2014) 129–141. doi:10.1016/j.cis.2014.02.004.

[12] A. Farutin, C. Misbah, Squaring, Parity Breaking, and S Tumbling of Vesicles under Shear Flow, Phys. Rev. Lett. 109 (24) (2012) 248106. doi:10.1103/PhysRevLett.109.248106.

[13] C. de Loubens, J. Deschamps, G. Boedec, M. Leonetti, Stretching of capsules in an elongation flow, a route to constitutive law, J. Fluid Mech. 767 (R3). doi:10.1017/jfm.2015.69.

[14] C. de Loubens, J. Deschamps, F. Edwards-Levy, M. Leonetti, Tank-treading of microcapsules in shear flow, J. Fluid Mech. 789 (2016) 750–767. doi:10.1017/jfm.2015.758.

[15] T. Omori, T. Ishikawa, D. Barthès-Biesel, A.-V. Salsac, Y. Imai, T. Yamaguchi, Tension of red blood cell membrane in simple shear flow, Phys. Rev. E 86 (5) (2012) 056321. doi:10.1103/PhysRevE.86.056321.

[16] G. Boedec, M. Jaeger, M. Leonetti, Settling of a vesicle in the limit of quasispherical shapes, J. Fluid Mech. 690 (2012) 227–261. doi:10.1017/jfm.2011.427.

[17] G. Boedec, M. Jaeger, M. Leonetti, Sedimentation-induced tether on a settling vesicle, Phys. Rev. E 88 (1) (2013) 010702. doi:10.1103/PhysRevE.88.010702.

[18] I. R. Suárez, C. Leidy, G. Téllez, G. Gay, A. Gonzalez-Mancera, Slow Sedimentation and Deformability of Charged Lipid Vesicles, PLoS ONE 8 (7) (2013) e68309. doi:10.1371/journal.pone.0068309.

[19] H.-H. Boltz, J. Kierfeld, Shapes of sedimenting soft elastic capsules in a viscous fluid, Phys. Rev. E 92 (3) (2015) 033003. doi:10.1103/PhysRevE.92.033003.

[20] H. Zhao, A. H. Isfahani, L. N. Olson, J. B. Freund, A spectral boundary integral method for flowing blood cells, J. Comput. Phys. 229 (10) (2010) 3726–3744. doi:10.1016/j.jcp.2010.01.024.

[21] J. B. Freund, The flow of red blood cells through a narrow spleen-like slit, Phys. Fluids 1994-Present 25 (11) (2013) 110807. doi:10.1063/1.4819341.

[22] R. Kusters, T. van der Heijden, B. Kaoui, J. Harting, C. Storm, Forced transport of deformable containers through narrow constrictions, Phys. Rev. E 90 (3) (2014) 033006. doi:10.1103/PhysRevE.90.033006.

[23] A. Daddi-Moussa-Ider, A. Guckenberger, S. Gekle, Long-lived anomalous thermal diffusion induced by elastic cell membranes on nearby particles, Phys. Rev. E 93 (1) (2016) 012612. doi:10.1103/PhysRevE.93.012612.

[24] R. Fåhræus, T. Lindqvist, The viscosity of the blood in narrow capillary tubes, Am. J. Physiol. 96 (3) (1931) 562–568.

[25] A. R. Pries, T. W. Secomb, T. Gessner, M. B. Sperandio, J. F. Gross, P. Gaehtgens, Resistance to blood flow in microvessels in vivo., Circ. Res. 75 (5) (1994) 904–915. doi:10.1161/01.RES.75.5.904.

[26] A. R. Pries, T. W. Secomb, P. Gaehtgens, Biophysical aspects of blood flow in the microvasculature, Cardiovasc. Res. 32 (4) (1996) 654–667. doi:10.1016/S0008-6363(96)00065-X.

[27] D. A. Fedosov, W. Pan, B. Caswell, G. Gompper, G. E. Karniadakis, Predicting human blood viscosity in silico, PNAS 108 (29) (2011) 11772–11777. doi:10.1073/pnas.1101210108.

[28] T. Krüger, Computer simulation study of collective phenomena in dense suspensions of red blood cells under shear, Ph.D. thesis, Ruhr-Universität Bochum (2012). doi:10.1007/978-3-8348-2376-2.

[29] T. Krüger, B. Kaoui, J. Harting, Interplay of inertia and deformability on rheological properties of a suspension of capsules, J. Fluid Mech. 751 (2014) 725–745. doi:10.1017/jfm.2014.315.

[30] P. A. Aarts, S. A. van den Broek, G. W. Prins, G. D. Kuiken, J. J. Sixma, R. M. Heethaar, Blood platelets are concentrated near the wall and red blood cells, in the center in flowing blood., Arterioscler. Thromb. Vasc. Biol. 8 (6) (1988) 819–824. doi:10.1161/01.ATV.8.6.819.

[31] J. B. Freund, M. M. Orescanin, Cellular flow in a small blood vessel, J. Fluid Mech. 671 (2011) 466–490. doi:10.1017/S0022112010005835.

[32] H. Zhao, E. S. G. Shaqfeh, V. Narsimhan, Shear-induced particle migration and margination in a cellular suspension, Phys. Fluids 1994-Present 24 (1) (2012) 011902. doi:10.1063/1.3677935.
[33] D. Katanov, G. Gompper, D. A. Fedosov, Microvascular blood flow resistance: Role of red blood cell migration and dispersion, Microvasc. Res. 99 (2015) 57–66. doi:10.1016/j.mvr.2015.02.006.
[34] J. B. Freund, Leukocyte margination in a model microvessel, Phys. Fluids 1994-Present 19 (2) (2007) 023301. doi:10.1063/1.2472479.
[35] K. Vahidkhah, S. L. Diamond, P. Bagchi, Platelet Dynamics in Three-Dimensional Simulation of Whole Blood, Biophys. J. 106 (11) (2014) 2529–2540. doi:10.1016/j.bpj.2014.04.028.
[36] K. Müller, D. A. Fedosov, G. Gompper, Margination of micro- and nano-particles in blood flow and its effect on drug delivery, Sci. Rep. 4. doi:10.1038/srep04871.
[37] D. A. Fedosov, G. Gompper, White blood cell margination in microcirculation, Soft Matter 10 (17) (2014) 2961–2970. doi:10.1039/C3SM52860J.
[38] S. Fitzgibbon, A. P. Spann, Q. M. Qi, E. S. G. Shaqfeh, In Vitro Measurement of Particle Margination in the Microchannel Flow: Effect of Varying Hematocrit, Biophys. J. 108 (10) (2015) 2601–2608. doi:10.1016/j.bpj.2015.04.013.
[39] S. Gekle, Strongly Accelerated Margination of Active Particles in Blood Flow, Biophys. J. 110 (2) (2016) 514–520. doi:10.1016/j.bpj.2015.12.005.
[40] K. Müller, D. A. Fedosov, G. Gompper, Understanding particle margination in blood flow – A step toward optimized drug delivery systems, Med. Eng. Phys. 38 (1) (2016) 2–10. doi:10.1016/j.medengphy.2015.08.009.
[41] M. Mehrabadi, D. N. Ku, C. K. Aidun, Effects of shear rate, confinement, and particle parameters on margination in blood flow, Phys. Rev. E 93 (2) (2016) 023109. doi:10.1103/PhysRevE.93.023109.
[42] G. K. Youngren, A. Acrivos, Stokes flow past a particle of arbitrary shape: a numerical method of solution, J. Fluid Mech. 69 (2) (1975) 377–403. doi:10.1017/S0022112075001486.
[43] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, no. 8 in Cambridge Texts in Applied Mathematics, Cambridge University Press, New York, 1992. doi:10.1017/CBO9780511624124.
[44] C. Pozrikidis, Interfacial Dynamics for Stokes Flow, J. Comput. Phys. 169 (2) (2001) 250–301. doi:10.1006/jcph.2000.6582.
[45] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Numerical Mathematics and Scientific Computation, Clarendon Press, Oxford, 2001.
[46] B. Dünweg, A. J. C. Ladd, Lattice Boltzmann Simulations of Soft Matter Systems, in: P. C. Holm, P. K. Kremer (Eds.), Advanced Computer Simulation Approaches for Soft Matter Sciences III, no. 221 in Advances in Polymer Science, Springer Berlin Heidelberg, 2009, pp. 89–166. doi:10.1007/978-3-540-87706-6_2.
[47] C. K. Aidun, J. R. Clausen, Lattice-Boltzmann Method for Complex Flows, Annu. Rev. Fluid Mech. 42 (1) (2010) 439–472. doi:10.1146/annurev-fluid-121108-145519.
[48] T. Krüger, F. Varnik, D. Raabe, Efficient and accurate simulations of deformable particles immersed in a fluid using a combined immersed boundary lattice Boltzmann finite element method, Comput. Math. Appl. 61 (12) (2011) 3485–3505. doi:10.1016/j.camwa.2010.03.057.
[49] H. Noguchi, G. Gompper, Fluid Vesicles with Viscous Membranes in Shear Flow, Phys. Rev. Lett. 93 (25) (2004) 258102. doi:10.1103/PhysRevLett.93.258102.
[50] H. Noguchi, G. Gompper, Dynamics of fluid vesicles in shear flow: Effect of membrane viscosity and thermal fluctuations, Phys. Rev. E 72 (1) (2005) 011901. doi:10.1103/PhysRevE.72.011901.
[51] G. Gompper, T. Ihle, D. M. Kroll, R. G. Winkler, Multi-Particle Collision Dynamics: A Particle-Based Mesoscale Simulation Approach to the Hydrodynamics of Complex Fluids, in: C. Holm, K. Kremer (Eds.), Advanced Computer Simulation Approaches for Soft Matter Sciences III, no. 221 in Advances in Polymer Science, Springer, Berlin, Heidelberg, 2009, pp. 1–87. doi:10.1007/978-3-540-87706-6_1.
[52] D. A. Fedosov, B. Caswell, G. E. Karniadakis, Dissipative particle dynamics modeling of red blood cells, in: Computational Hydrodynamics of Capsules and Biological Cells, Chapman & Hall/CRC Mathematical and Computational Biology Series, CRC Press, 2010, pp. 183–218. doi:10.1201/EBK1439820056-c6.
[53] D. Barthès-Biesel, Motion and Deformation of Elastic Capsules and Vesicles in Flow, Annu. Rev. Fluid Mech. 48 (1) (2016) 25–52. doi:10.1146/annurev-fluid-122414-034345.
[54] R. Skalak, A. Tozeren, R. P. Zarda, S. Chien, Strain Energy Function of Red Blood Cell Membranes, Biophys. J. 13 (3) (1973) 245–264. doi:10.1016/S0006-3495(73)85983-1.
[55] T. Omori, T. Ishikawa, D. Barthès-Biesel, A.-V. Salsac, J. Walter, Y. Imai, T. Yamaguchi, Comparison between spring network models and continuum constitutive laws: Application to the large deformation of a capsule in shear flow, Phys. Rev. E 83 (4) (2011) 041918. doi:10.1103/PhysRevE.83.041918.
[56] P. B. Canham, The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell, J. Theor. Biol. 26 (1) (1970) 61–81. doi:10.1016/S0022-5193(70)80032-7.
[57] W. Helfrich, Elastic Properties of Lipid Bilayers: Theory and Possible Experiments, Z. Naturforsch. C 28 (1973) 693–703.
[58] G. Boedec, M. Leonetti, M. Jaeger, 3D vesicle dynamics simulations with a linearly triangulated surface, J. Comput. Phys. 230 (4) (2011) 1020–1034. doi:10.1016/j.jcp.2010.10.021.
[59] A. Walter, H. Rehage, H. Leonhard, Shear induced deformation of microcapsules: shape oscillations and membrane folding, Coll. Surf. A. 183–185 (2001) 123–132. doi:10.1016/S0927-7757(01)00564-7.
[60] H. Luo, C. Pozrikidis, Buckling of a pre-compressed or pre-stretched membrane in shear flow, Int. J. Solids Struct. 44 (24) (2007) 8074–8085. doi:10.1016/j.ijsolstr.2007.05.027.
[61] X. Li, K. Sarkar, Front tracking simulation of deformation and buckling instability of a liquid capsule enclosed by an elastic membrane, J. Comput. Phys. 227 (10) (2008) 4998–5018. doi:10.1016/j.jcp.2008.01.034.

[62] R. Finken, S. Kessler, U. Seifert, Micro-capsules in shear flow, J. Phys.: Condens. Matter 23 (18) (2011) 184113. doi:10.1088/0953-8984/23/18/184113.
[63] C. Dupont, A.-V. Salsac, D. Barthès-Biesel, M. Vidrascu, P. L. Tallec, Influence of bending resistance on the dynamics of a spherical capsule in shear flow, Phys. Fluids 1994-Present 27 (5) (2015) 051902. doi:10.1063/1.4921247.
[64] E. Lac, D. Barthès-Biesel, N. A. Pelekasis, J. Tsamopoulos, Spherical capsules in three-dimensional unbounded Stokes flows: effect of the membrane constitutive law and onset of buckling, J. Fluid Mech. 516 (2004) 303–334. doi:10.1017/S002211200400062X.
[65] C. Pozrikidis, Resting shape and spontaneous membrane curvature of red blood cells, Math. Med. Biol. 22 (1) (2005) 34–52. doi:10.1093/imammb/dqh021.
[66] T. Biben, A. Farutin, C. Misbah, Three-dimensional vesicles under shear flow: Numerical study of dynamics and phase diagram, Phys. Rev. E 83 (3) (2011) 031921. doi:10.1103/PhysRevE.83.031921.
[67] A. Yazdani, P. Bagchi, Three-dimensional numerical simulation of vesicle dynamics using a front-tracking method, Phys. Rev. E 85 (5) (2012) 056308. doi:10.1103/PhysRevE.85.056308.
[68] A. Farutin, T. Biben, C. Misbah, 3D numerical simulations of vesicle and inextensible capsule dynamics, J. Comput. Phys. 275 (2014) 539–568. doi:10.1016/j.jcp.2014.07.008.
[69] K.-i. Tsubota, Short note on the bending models for a membrane in capsule mechanics: Comparison between continuum and discrete models, J. Comput. Phys. 277 (2014) 320–328. doi:10.1016/j.jcp.2014.08.007.
[70] J. Gounley, G. Boedec, M. Jaeger, M. Leonetti, Influence of surface viscosity on droplets in shear flow, J. Fluid Mech. 791 (2016) 464–494. doi:10.1017/jfm.2016.39.
[71] W.-X. Huang, C. B. Chang, H. J. Sung, Three-dimensional simulation of elastic capsules in shear flow by the penalty immersed boundary method, J. Comput. Phys. 231 (8) (2012) 3340–3364. doi:10.1016/j.jcp.2012.01.006.
[72] A. P. Spann, H. Zhao, E. S. G. Shaqfeh, Loop subdivision surface boundary integral method simulations of vesicles at low reduced volume ratio in shear and extensional flow, Phys. Fluids 1994-Present 26 (3) (2014) 031902. doi:10.1063/1.4869307.
[73] C. T. Loop, Smooth Subdivision Surfaces Based on Triangles, Master thesis, University of Utah, Utah, USA (1987).
[74] F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: a new paradigm for thin-shell finite-element analysis, Int. J. Numer. Methods Eng. 47 (12) (2000) 2039–2072. doi:10.1002/(SICI)1097-0207(20000430)47:12<2039::AID-NME872>3.0.CO;2-1.
[75] D. V. Le, J. White, J. Peraire, K. M. Lim, B. C. Khoo, An implicit immersed boundary method for three-dimensional fluid–membrane interactions, J. Comput. Phys. 228 (22) (2009) 8427–8445. doi:10.1016/j.jcp.2009.08.018.
[76] J. Walter, A.-V. Salsac, D. Barthès-Biesel, P. Le Tallec, Coupling of finite element and boundary integral methods for a capsule in a Stokes flow, Int. J. Numer. Methods Eng. 83 (7) (2010) 829–850. doi:10.1002/nme.2859.
[77] S. K. Veerapaneni, A. Rahimian, G. Biros, D. Zorin, A fast algorithm for simulating vesicle flows in three dimensions, J. Comput. Phys. 230 (14) (2011) 5610–5634. doi:10.1016/j.jcp.2011.03.045.
[78] O.-Y. Zhong-Can, W. Helfrich, Bending energy of vesicle membranes: General expressions for the first, second, and third variation of the shape energy and applications to spheres and cylinders, Phys. Rev. A 39 (10) (1989) 5280–5288. doi:10.1103/PhysRevA.39.5280.
[79] A. Laadhari, C. Misbah, P. Saramito, On the equilibrium equation for a generalized biological membrane energy by using a shape optimization approach, Phys. Nonlinear Phenom. 239 (16) (2010) 1567–1572. doi:10.1016/j.physd.2010.04.001.
[80] G. Xu, Convergence of discrete Laplace-Beltrami operators over surfaces, Comput. Math. Appl. 48 (3-4) (2004) 347–360. doi:10.1016/j.camwa.2004.05.001.
[81] G. Xu, Discrete Laplace–Beltrami operators and their convergence, Comput. Aided Geom. Des. 21 (8) (2004) 767–784. doi:10.1016/j.cagd.2004.07.007.
[82] T. D. Gatzke, C. M. Grimm, Estimating curvature on triangular meshes, Int. J. Shape Model. 12 (01) (2006) 1–28. doi:10.1142/S0218654306000810.
[83] M. Wardetzky, S. Mathur, F. Kälberer, E. Grinspun, Discrete Laplace Operators: No Free Lunch, in: A. Belyaev, M. Garland (Eds.), SGP '07 Proceedings of the fifth Eurographics symposium on Geometry processing, SGP '07, Eurographics Association, Barcelona, 2007, pp. 33–37.
[84] M. Belkin, J. Sun, Y. Wang, Discrete Laplace Operator on Meshed Surfaces, in: SCG '08: Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry, ACM, Maryland, 2008, pp. 278–287. doi:10.1145/1377676.1377725.
[85] M. Alexa, M. Wardetzky, Discrete Laplacians on General Polygonal Meshes, in: SIGGRAPH '11, ACM, New York, 2011, pp. 102:1–102:10. doi:10.1145/1964921.1964997.
[86] M. M. Mesmoudi, L. D. Floriani, P. Magillo, Concentrated Curvature for Mean Curvature Estimation in Triangulated Surfaces, in: M. Ferri, P. Frosini, C. Landi, A. Cerri, B. D. Fabio (Eds.), Computational Topology in Image Context, no. 7309 in Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 79–87. doi:10.1007/978-3-642-30238-1_9.
[87] X. Li, G. Xu, Y. J. Zhang, Localized discrete Laplace–Beltrami operator over triangular mesh, Comput. Aided Geom. Des. 39 (2015) 67–82. doi:10.1016/j.cagd.2015.09.001.
[88] Y. Kantor, D. R. Nelson, Phase transitions in flexible polymeric surfaces, Phys. Rev. A 36 (8) (1987) 4020–4032. doi:10.1103/PhysRevA.36.4020.
[89] G. Gompper, D. M. Kroll, Random Surface Discretizations and the Renormalization of the Bending Rigidity, J. Phys. Fr. 6 (10) (1996) 1305–1320. doi:10.1051/jp1:1996246.
[90] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, in: H.-C. Hege, K. Polthier (Eds.), Visualization and Mathematics III, no. III in Mathematics and Visualization, Springer, Berlin, Heidelberg, 2003, pp. 35–57. doi:10.1007/978-3-662-05105-4_2.

[91] B. Lévy, Laplace-Beltrami Eigenfunctions Towards an Algorithm That "Understands" Geometry, in: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006, SMI '06, IEEE Computer Society, Washington, DC, USA, 2006. doi:10.1109/SMI.2006.21.
[92] S. Komura, K. Tamura, T. Kato, Buckling of spherical shells adhering onto a rigid substrate, Eur. Phys. J. E 18 (3) (2005) 343–358. doi:10.1140/epje/e2005-00038-5.
[93] J. Li, M. Dao, C. T. Lim, S. Suresh, Spectrin-Level Modeling of the Cytoskeleton and Optical Tweezers Stretching of the Erythrocyte, Biophys. J. 88 (5) (2005) 3707–3719. doi:10.1529/biophysj.104.047332.
[94] I. V. Pivkin, G. E. Karniadakis, Accurate Coarse-Grained Modeling of Red Blood Cells, Phys. Rev. Lett. 101 (11) (2008) 118105. doi:10.1103/PhysRevLett.101.118105.
[95] D. A. Fedosov, B. Caswell, G. E. Karniadakis, A Multiscale Red Blood Cell Model with Accurate Mechanics, Rheology, and Dynamics, Biophys. J. 98 (10) (2010) 2215–2225. doi:10.1016/j.bpj.2010.02.002.
[96] H. Noguchi, G. Gompper, Shape transitions of fluid vesicles and red blood cells in capillary flows, PNAS 102 (40) (2005) 14159–14164. doi:10.1073/pnas.0504243102.
[97] H. Noguchi, G. Gompper, Swinging and Tumbling of Fluid Vesicles in Shear Flow, Phys. Rev. Lett. 98 (12) (2007) 128103. doi:10.1103/PhysRevLett.98.128103.
[98] J. L. McWhirter, H. Noguchi, G. Gompper, Flow-induced clustering and alignment of vesicles and red blood cells in microcapillaries, PNAS 106 (15) (2009) 6039–6043. doi:10.1073/pnas.0811484106.
[99] J. L. McWhirter, H. Noguchi, G. Gompper, Deformation and clustering of red blood cells in microcapillary flows, Soft Matter 7 (22) (2011) 10967–10977. doi:10.1039/C1SM05794D.
[100] J. L. McWhirter, H. Noguchi, G. Gompper, Ordering and arrangement of deformed red blood cells in flow through microcapillaries, New J. Phys. 14 (8) (2012) 085026. doi:10.1088/1367-2630/14/8/085026.
[101] S. Jin, R. R. Lewis, D. West, A comparison of algorithms for vertex normal computation, Vis. Comput. 21 (1-2) (2005) 71–82. doi:10.1007/s00371-004-0271-1.
[102] A. Yazdani, P. Bagchi, Influence of membrane viscosity on capsule dynamics in shear flow, J. Fluid Mech. 718 (2013) 569–595. doi:10.1017/jfm.2012.637.
[103] M. Loewenberg, E. J. Hinch, Numerical simulation of a concentrated emulsion in shear flow, J. Fluid Mech. 321 (1996) 395–419. doi:10.1017/S002211209600777X.
[104] A. Z. Zinchenko, M. A. Rother, R. H. Davis, A novel boundary-integral algorithm for viscous interaction of deformable drops, Phys. Fluids 1994-Present 9 (6) (1997) 1493–1511. doi:10.1063/1.869275.
[105] I. B. Bazhlekov, P. D. Anderson, H. E. H. Meijer, Nonsingular boundary integral method for deformable drops in viscous flows, Phys. Fluids 1994-Present 16 (4) (2004) 1064–1081. doi:10.1063/1.1648639.
[106] C. Zhang, T. Chen, Efficient feature extraction for 2D/3D objects in mesh representation, in: Proceedings ICIP 2001, Vol. 3, IEEE Signal Processing Society, Thessaloniki, 2001, pp. 935–938. doi:10.1109/ICIP.2001.958278.
[107] R. Aiello, F. Banterle, N. Pietroni, L. Malomo, P. Cignoni, R. Scopigno, Compression and Querying of Arbitrary Geodesic Distances, in: V. Murino, E. Puppo (Eds.), Image Analysis and Processing — ICIAP 2015, no. 9279 in Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 282–293. doi:10.1007/978-3-319-23231-7_26.
[108] F. R. Cunha, M. Loewenberg, A study of emulsion expansion by a boundary integral method, Mech. Res. Commun. 30 (6) (2003) 639–649. doi:10.1016/S0093-6413(03)00068-5.
[109] Y. Saad, M. Schultz, GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, SIAM J. Sci. and Stat. Comput. 7 (3) (1986) 856–869. doi:10.1137/0907058.
[110] E. Evans, Y.-C. Fung, Improved measurements of the erythrocyte geometry, Microvasc. Res. 4 (4) (1972) 335–347. doi:10.1016/0026-2862(72)90069-6.
[111] D.-V. Le, Subdivision elements for large deformation of liquid capsules enclosed by thin shells, Comput. Methods Appl. Mech. Eng. 199 (37-40) (2010) 2622–2632. doi:10.1016/j.cma.2010.04.014.
[112] M. C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques, Int. J. Numer. Meth. Engng. 20 (4) (1984) 745–756. doi:10.1002/nme.1620200412.
[113] D.-V. Le, S. T. Wong, A front-tracking method with Catmull–Clark subdivision surfaces for studying liquid capsules enclosed by thin shells in shear flow, J. Comput. Phys. 230 (9) (2011) 3538–3555. doi:10.1016/j.jcp.2011.01.047.
[114] Z. Y. Luo, S. Q. Wang, L. He, F. Xu, B. F. Bai, Inertia-dependent dynamics of three-dimensional vesicles and red blood cells in shear flow, Soft Matter 9 (40) (2013) 9651–9660. doi:10.1039/C3SM51823J.
[115] K. S. Chang, W. L. Olbricht, Experimental studies of the deformation and breakup of a synthetic capsule in steady and unsteady simple shear flow, J. Fluid Mech. 250 (1993) 609–633. doi:10.1017/S0022112093001582.
[116] S. Ramanujan, C. Pozrikidis, Deformation of liquid capsules enclosed by elastic membranes in simple shear flow: large deformations and the effect of fluid viscosities, J. Fluid Mech. 361 (1998) 117–143. doi:10.1017/S0022112098008714.
[117] G. R. Cowper, Gaussian quadrature formulas for triangles, Int. J. Numer. Methods Eng. 7 (3) (1973) 405–408. doi:10.1002/nme.1620070316.
[118] C. Pozrikidis, Finite deformation of liquid capsules enclosed by elastic membranes in simple shear flow, J. Fluid Mech. 297 (1995) 123–152. doi:10.1017/S002211209500303X.
[119] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition, Cambridge University Press, Cambridge, 2002.
[120] A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Roehm, P. Košovan, C. Holm, ESPResSo 3.1: Molecular Dynamics Software for Coarse-Grained Models, in: M. Griebel, M. A. Schweitzer (Eds.), Meshfree Methods for Partial Differential Equations VI, no. 89 in Lecture Notes in Computational Science and Engineering, Springer, Berlin, Heidelberg, 2013, pp. 1–23. doi:10.1007/978-3-642-32979-1_1.
[121] H. J. Limbach, A. Arnold, B. A. Mann, C. Holm, ESPResSo—an extensible simulation package for research on soft

matter systems, Comput. Phys. Commun. 174 (9) (2006) 704–727. doi:10.1016/j.cpc.2005.10.005.
[122] P. R. Zarda, S. Chien, R. Skalak, Elastic deformations of red blood cells, J. Biomech. 10 (4) (1977) 211–221. doi:10.1016/0021-9290(77)90044-6.