MeMC: A package for monte-carlo simulations of spherical shells

XX April 2021

Summary

The MeMC is an open-source software package for monte-carlo simulation of elastic shells. It is designed as a tool to interpret the force-distance data generated by indentation of biological nano-vesicles by atomic force microscopes. The code is written in c++ and python. The code is customizable – new modules can be added in a straightforward manner.

Statement of need and purpose of software

Micro and nano vesicles, both natural and synthetic, play a crucial role in biology and medicine. The physical properties of these vesicles play an important role in their biological functions [@phillips2012physical]. Hence it is important to be able to measure their elastic constants, in particular the Young's modulus and the bending rigidity. One way to measure the elastic constants of biological objects, e.g., a red blood cell (RBC), is to poke them with an atomic force microscope (AFM) to obtain a force-distance curve. Then we must model the biological object as an elastic material and by fitting this model to the experimental force-distance curve estimate the parameters of the elastic model, i.e., the elastic constants. As an example, consider a force-distance curve obtained by AFM measurement of a RBC. The RBC is modeled as a linear elastic material with a Young's modulus, $Y_{3\mathrm{d}}$. Typically a Hertzian model of elastic bodies in contact [@LLelast section 9] is used to measure $Y_{3\mathrm{d}}$. Nano vesicles differ from (micro-meter scale) cells in two important ways

1. The nano-vesicles are much smaller hence thermal fluctuations may effectively renormalize the elastic coefficients [@paulose2012fluctuating; @kovsmrlj2017statistical].
2. Cell membranes are strongly coupled to an underlying cytoskelton. Hence they may be modeled by a solid body [@hw2002stomatocyte] but nano-vesicles must be modeled as liquid filled elastic membranes.

Hence, to be able to interpret the force-distance curve of nano-vesicles we need to solve for the elastic response of a thermally fluctuating elastic shell.

There are commercial packages, e.g., COMSOL [@comsol], to calculate the force-distance curve of solid bodies and closed membranes with fluids inside under the action of external forces but to the best of our knowledge there is no package that includes thermal effects. Monte-carlo simulations of elastic membranes, that does include thermal fluctuations, have been done for more than three decades [@goetz1999mobility; @bowick2001universality; @auth2005fluctuation; @paulose2012fluctuating], see also [@gompper2004triangulated for a review]. But to the best of our knowledge there are no open-source code available. The goal of this package is to fill this gap in open-source software.

Theoretical background

Our model of nano-vesicles is an amorphous membrane enclosing an incompressible fluid[@vorselen2017competition]. Unlike a solid ball, the force-distance relationship for such a model is linear for small deformation [@vorselen2017competition; @paulose2012fluctuating] if we ignore thermal fluctuations. Ref. [@vorselen2017competition] uses a similar model, ignoring thermal fluctuations, to interpret AFM measurement of nano-vesicles.

Let us consider a (three dimensional) material with Young's modulus $Y_{3\mathrm{d}}$ and Poisson's ratio $\sigma_{3\mathrm{d}}$ and make a membrane out of it. Then the bending modulus and the in-plane Young's modulus is given by [@LLelast section 13 and 14]

$$B = \frac{Y_{3\mathrm{d}} h^3}{12(1-\sigma_{3\mathrm{d}}^2)} \quad \text{and} \quad Y = Y_{3\mathrm{d}} h,$$

where h is the thickness of the membrane. This need not necessarily hold for biological membranes. Nevertheless consider a fluid enclosed in a solid membrane, as done in [@paulose2012fluctuating]. We consider an elastic energy

$$\begin{aligned}
\mathcal{E}\left[w, \mathbf{u}\right] &= \int d^2x \left[\frac{B}{2}\left(\nabla^2 w\right)^2 + \mu u_{ij}^2 + \frac{\lambda}{2} u_{kk}^2 - pw\right] \\
u_{ij} &= \frac{1}{2}\left(\partial_i u_j + \partial_j u_i + \partial_i w \partial_j w\right) - \delta_{ij}\frac{w}{R}
\end{aligned}$$

where w is the out-of-plane deformation of the shell, and $\mathbf{u}$ is the in-plane deformation, p is the pressure, λ and μ are the two in-plane Láme coefficients and B is the bending modulus. The Láme coefficients are related to other elastic constant as follows [@LLelast]

$$K = \lambda + \frac{2}{3}\mu$$
$$Y = \frac{9K\mu}{3K+\mu}$$
$$\sigma = \frac{1}{2}\frac{3K-2\mu}{3K+\mu}$$

Here K is the volume compressibility, Y the Young's modulus, and σ the Poisson ratio.

If we consider the material to be incompressible, $K \to \infty$ and $\sigma = 1/2$, then there are two elastic constant, the bending rigidity B and the Young's modulus Y. Consequently, there are two dimensionless numbers, the the Föppl–von-Karman number

$$\mathrm{FvK} = \frac{YR^2}{B}$$

and the Elasto-Thermal number:

$$\mathrm{ET} = \frac{k_\mathrm{B}T}{B}\sqrt{\mathrm{FvK}}$$

where R is the radius of the spherical shell.

Using values of Y and B from molecular dynamics simulations of lipid bilayers [@boek2005mechanical], $Y = 1.7\mathrm{N/m}$ and $B = 5k_\mathrm{B}T$ and $R = 100$ nano meter, we have $\mathrm{FvK} \approx 0.3 \times 10^7$ which is close to the Föppl–von-Karman number for Graphene sheets and $\mathrm{ET} \approx 10^3$.

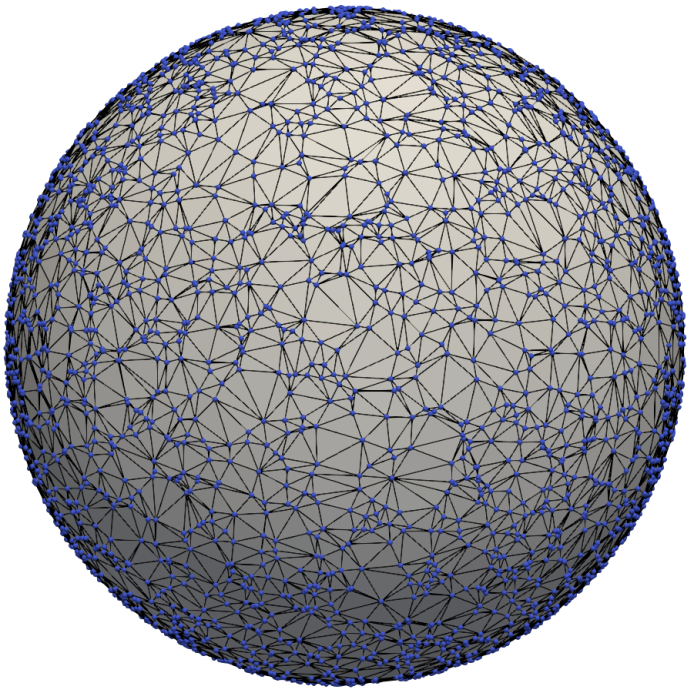
Numerical implementation

Grid

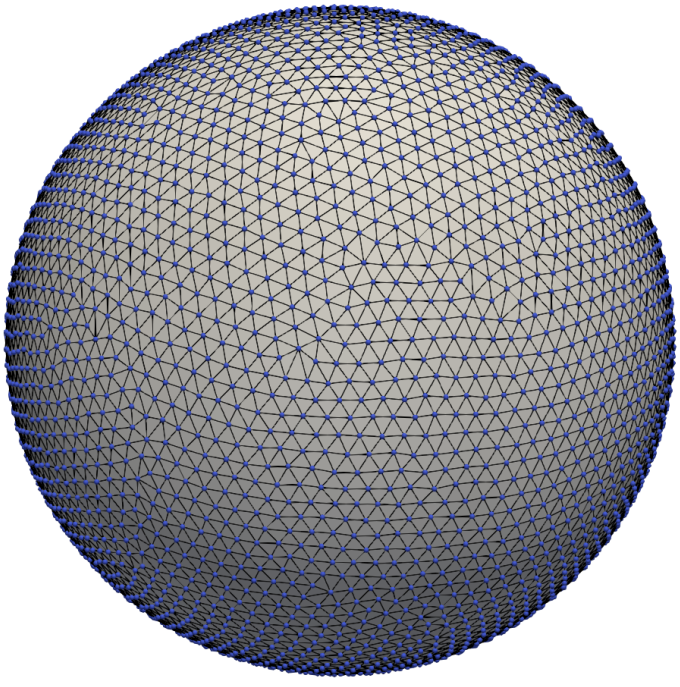
Following Ref. [@gompper2004triangulated], we use a triangulated-network grid in the following manner. We start with N randomly chosen points on a sphere . Then, we run a Monte-Carlo simulation, with a Lenard-Jones (LJ) repelling potential, of these points moving on the surface. Once the surface Monte-Carlo (SMC) has reached an equilibrium, we use the algorithm in Ref. [@caroli2009robust] to construct the Delaunay triangulation of these points. The connection between the points thus formed is kept unchanged. In the rest of this paper we call this the *initial configuration*. A different snapshot from the same equilibrium gives an equivalent but differently triangulated grid.

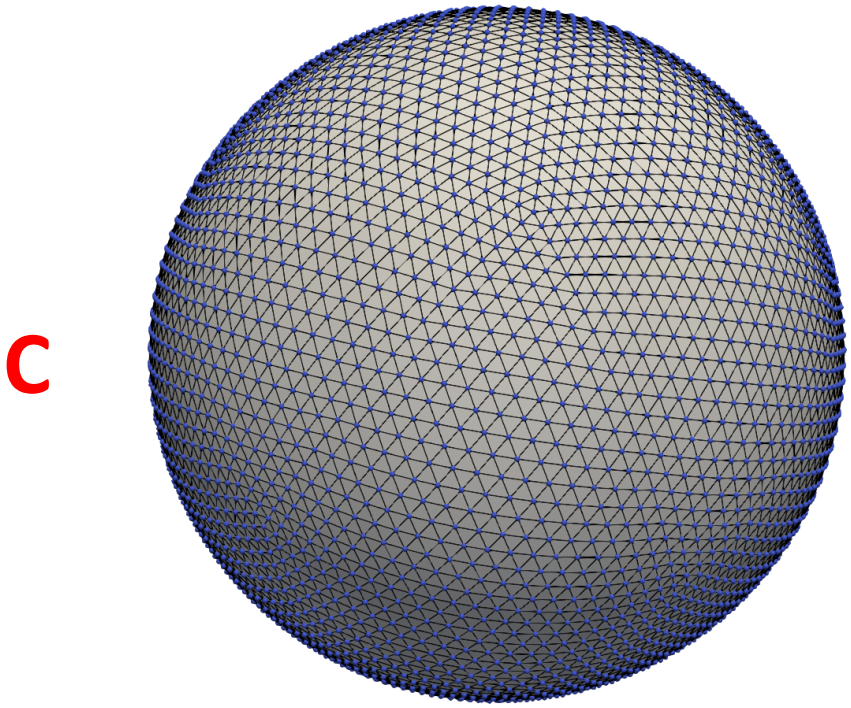
An alternative is to use a regular grid [@vliegenthart2006mechanical; @buenemann2008elastic]. This is achieved by approximating the sphere with geodesic polyhedron (C) . They are available from the *Meshzoo* library [@meshzoo]. In this paper we use $N = 5120$ for the random grid and $N = 5292$ for the regular grid.

A



B





Energy

The basic algorithm of Monte-Carlo simulations is straightforward and well-known [see, e.g., @baumgartner2013applications]. We randomly choose a point on the grid and move it by a random amount. We calculate the change in energy due to this movement. We accept or reject the move by the standard Metropolis algorithm. In our code the energy has the following contributions

$$E = E_{\mathrm{s}} + E_{\mathrm{bend}} + E_{\mathrm{bulk}}$$

where E_{s} is the contribution from stretching, E_{bend} is the contribution from bending, and E_{bulk} is the contribution from the bulk modulus. We describe each of these in turn.

Stretching

In the initial configuration, two neighboring points with coordinates R_{i} and R_{j} are connected by a bond of length ℓ^0_{ij}. When the i-th point is moved all its bonds with the neighbors change from their initial length. We model each of these bonds by a harmonic spring and calculate the stretching energy by

$$E_{\mathrm{s}} = \frac{1}{2}\sum_{i}\frac{H}{2}\sum_{j(i)}\left(R_{\mathrm{ij}} - \ell^{0}_{\mathrm{ij}}\right)^{2} \quad \text{where}$$
$$\mathbf{R}_{\mathrm{ij}} \equiv |R_{\mathrm{i}} - R_{\mathrm{j}}|.$$

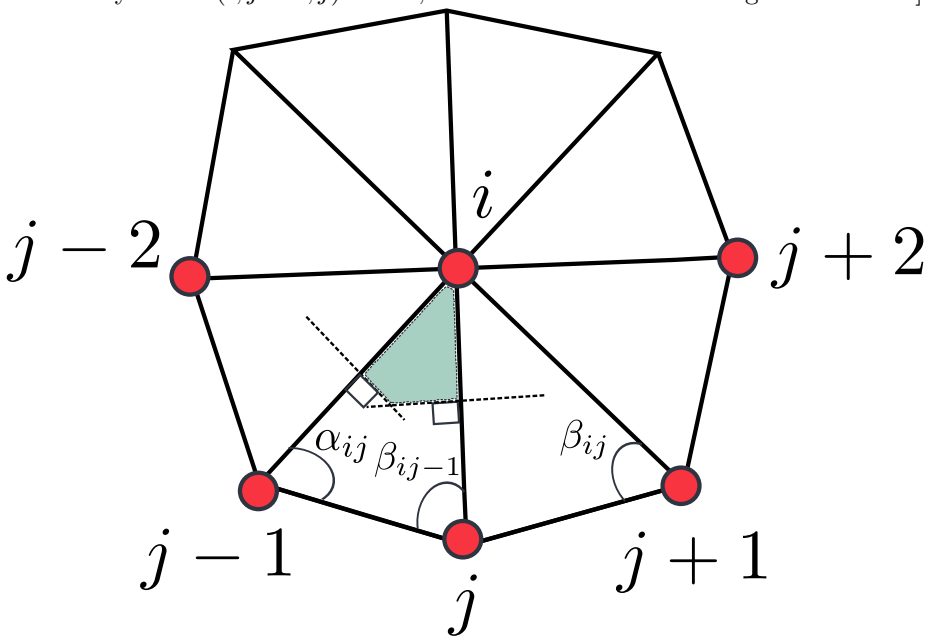
Here the notation $j(i)$ denotes that the sum is over all the nearest neighbors of the i-the point.

The Láme coefficients and the Young's modulus are given by [@seung1988defects]

$$\lambda = \mu = \frac{\sqrt{3}}{4}H, \quad Y = \frac{2}{\sqrt{3}}H$$

Bending

[An example of triangulated mesh at the node i. $\alpha_{\mathrm{ij}}, \beta_{\mathrm{ij}}$ are the angles opposite to the bond ij. Shaded part is the Voronoi region of triangle T defined by nodes $(i, j-1, j)$. Here, we consider that the triangle T is acute.]



To calculate the bending contribution we need to calculate the curvature. In the continuum limit, $N \to \infty$, bending energy [@nelson2004statistical] is:

$$E_{\mathrm{B}} = \frac{B}{2}\int (\nabla^{2}\mathbf{R})^{2} dS,$$

where $\nabla^{2}\mathbf{R}$ is Laplacian of $\mathbf{R}$ on the surface of the sphere.

A general introduction to discretization of Laplacian on a triangulated mesh is given in Refs.[@itzykson1986proceedings; @hege2003visualization]. Laplacian on a 2D manifold embedded in $\mathbb{R}^3$ is:

$$\mathbf{L}(\mathbf{R}) = 2\kappa(\mathbf{R})\hat{m}(\mathbf{R}),$$

where $\kappa(\mathbf{R})$ is the mean curvature, and $\hat{m}(\mathbf{R})$ is the normal to the surface at $\mathbf{R}$. Note that, $\hat{m}(\mathbf{R})$ is a local property of a point P with coordinates $\mathbf{R}$ and it is not necessarily the outward normal of the closed surface. In the discrete form [@hege2003visualization; @meyer2003discrete], $\mathcal{A}_i$ is the area of Voronoi dual cell at the node i, and $\alpha_{\mathrm{ij}}, \beta_{\mathrm{ij}}$ are the angles opposite to bond ij as shown in . Consider the triangle T in defined by its vertices $(i, j-1, j)$. If T is non-obtuse its circumcenter lies within it, hence so does the Voronoi region. Let $\mathcal{A}^{\mathrm{c}}$ be the area of shaded region in given by [@meyer2003discrete; @hege2003visualization],

$$\mathcal{A}^{\mathrm{c}} = \frac{1}{8}\left[R_{\mathrm{ij}}^2 \cot(\alpha_{\mathrm{ij}}) + R_{\mathrm{ij-1}}^2 \cot(\beta_{\mathrm{ij-1}})\right].$$

If there is an obtuse angle in triangle T, the Voronoi region is not necessarily enclosed by the triangle [@hege2003visualization]. For such cases, instead of $\mathcal{A}^{\mathrm{c}}$, we use $\mathcal{A}^{\mathrm{b}}$, defined as [@hege2003visualization; @meyer2003discrete]:

$$\mathcal{A}^{\mathrm{b}} = \left\{ \begin{array}{l} \frac{\mathrm{area(T)}}{2}, \text{angle of T at i is obtuse} \\ \frac{\mathrm{area(T)}}{4}, \text{any other angle is obtuse} \end{array} \right\},$$

where $\mathrm{area}(T) = 0.5\left|\mathbf{R}_{\mathrm{ij}} \times \mathbf{R}_{\mathrm{ij-1}}\right|$ is the area of the triangle T. The area $\mathcal{A}_i$ is obtained by summing up the contributions from all the triangles in , e.g., the contribution from the triangle T is the shaded area.

For a closed surface the bending energy must be calculated relative to the spontaneous curvature, i.e., its discretised form is

$$E_{\mathrm{B}} = \frac{B}{2}\mathcal{A}_i\left(\mathbf{L}_i - C\hat{n}\right)^2.$$

where C is the spontaneous curvature, for a sphere $C = 2/R$, where R is radius of the sphere and $\hat{n}$ is the outward normal to the surface. Hence not only the magnitude but also the vector nature of the surface Laplacian must be determined. For every triangle in the initial configuration, i.e., when all the points lie on the surface of a sphere, the outward unit normal can be calculated in a straightforward manner. For example, for the triangle T in it can be calculated by finding out the unit vector that points along $\mathbf{R}_{\mathrm{ij-1}} \times \mathbf{R}_{\mathrm{ij}}$. Hence, at any time, if we access the points around the node i in counterclockwise manner when viewed from outside we are guaranteed to obtain the outward normal. We ensure this by sorting appropriately the points around every node in the initial configuration. As the connectivity of the mesh remains unchanged this property is preserved at all future times.

To sort the neighbors around any node i, we rotate the coordinate system such that, the z axis passes through the point i along the vector $\mathbf{R}_i$. In this coordinate system we sort the neighbors by their azimuthal angle.

Note that unlike Ref. [@gompper2004triangulated] we do not incorporate self-avoidance.

Bulk

We assume that the liquid inside the vesicle is incompressible [1]. This is implemented by adding a energy cost to the volume change. At any point, the total contribution to the bulk energy is

$$E_{\text{bulk}} = K \left(\frac{V}{V_0} - 1 \right)^2 ,$$

where K is bulk modulus, V is current volume, and V_0 is the initial volume of the vesicle. As we move the point i by a random amount, the change in bulk energy is

$$\Delta E_{\text{bulk}} = 2K \frac{\Delta V (V - V_0)}{V_0^2} + \left(\frac{\Delta V}{V_0} \right)^2 ,$$

where ΔV is the change in volume due to the move.

Pressure

With addition of contribution from pressure difference from inside and outside the shell our code can also be used for pressurized shells.

Sticking to a solid surface

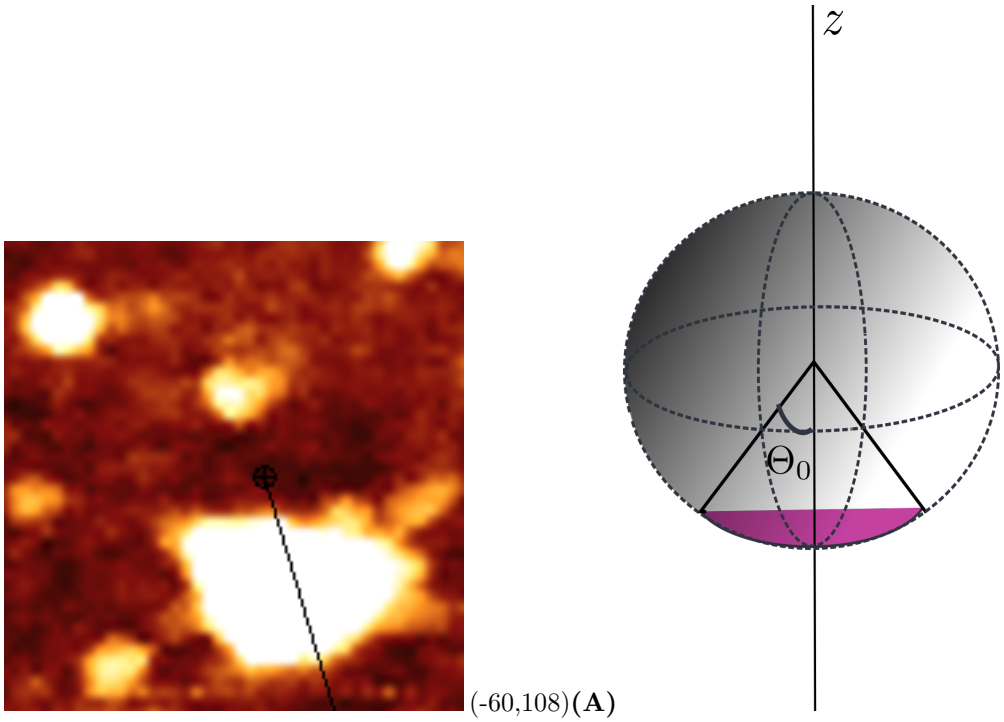
As a specific example of nano-vesicle, we consider an exosome. We quote from Ref. [@pegtel2019exosomes] " Exosomes are small, single-membrane, secreted organelles of $\sim$ 30 to $\sim$ 200 nm in diameter that have the same topology as the cell and are enriched in selected proteins, lipids, nucleic acids, and glycoconjugates. " The exosomes that we consider here were collected from immortalized cell line and extracted following the procedures as described in Ref. [@cavallaro2019label]. To measure the force-distance curve it is necessary to fix an exosome on a transparent coverslip. This was done by electrostatic coupling to a PLL coated surface by incubating them at room temperature for one hour, see Ref [@cavallaro2019label]. As an illustration, in , we show a typical experimental measurement of the height above a flat surface as measured by the AFM. After being stuck to the flat surface the free surface forms a spherical cap.

[1] This is different from assuming a semi-permeable membrane, as done in Ref. [@vorselen2017competition], in which case the liquid can flow in or out and the osmotic pressure of solutes decreases and increases accordingly.

To reproduce such experiments as closely as possibly we need to fix the vesicle to a flat surface. This is implemented by the Lenard-Jones potential:

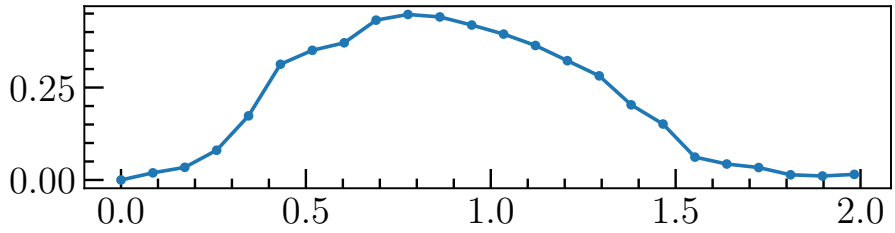
$$V_{\mathrm{LJ}}(r) \equiv 4\epsilon_{\mathrm{w}} \left[\left(\frac{\sigma_{\mathrm{w}}}{r} \right)^{12} - \left(\frac{\sigma_{\mathrm{w}}}{r} \right)^{6} \right]$$

What fraction of the vesicle is fixed to the flat surface is parametrized by the angle Θ_0 (see) which is a parameter in our code. We choose a system of coordinates with its origin at the center of the vesicle and the z axis pointing radially outward through the north pole. All the grid points on the surface whose polar angle is greater than Θ_0 are selected such that the sticking potential acts only on them, see .



(-60,108)**(A)**

(-60,90)**(B)**



(-210,50)**(C)**

AFM tip

To model the interaction between AFM tip and the vesicle, We model the shape of tip as paraboloid and we use only the repelling part of the Lenard-Jones potential:

$$V_{\mathrm{RLJ}}(r) \equiv 4\epsilon_{\mathrm{A}} \left(\frac{\sigma_{\mathrm{A}}}{r}\right)^{12}$$

For every point on the vesicle we calculate the shortest distance of this point to the paraboloid and use this distance as the argument of function V_{RLJ} in .

Dependencies

The code requires the following:

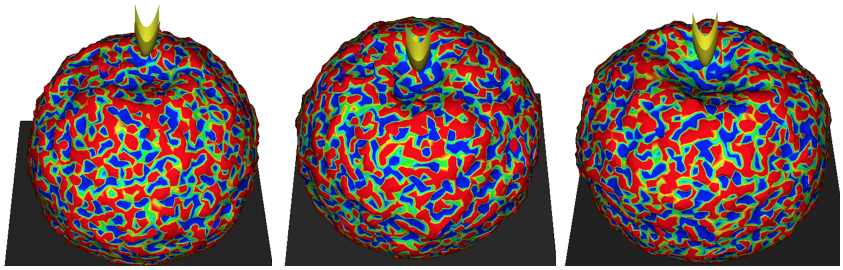
- A c++ compiler. We have tested the code against gnu g++ version 11.2.0 on x86_64 CPU.
- Hdf5 libraries for reading and writing data.
- Python version 3.8 (NOTE:: Cross check versions we use) with scipy, numpy, h5py and numpy-quaternion installed.
- For three-dimensional visualization we use VisIt [@HPV:VisIt].

Typical workflow and test

We have tested our code in LINUX operating system. We expect it to work without any problem in any similar environment. It may also work with WINDOWS although we have not tested this aspect.

The github repository [@memc] contains a a file named `README.md` that contains instruction to install and run the code. In we show three typical snapshots from our code for three different position of the AFM tip.

In the github repository, we also provide a subdirectory called `Examples`. By executing the shell script `execute.sh` in that directory the user can run the code (without the AFM tip and the bottom plate). It takes almost 30 minutes on Intel(R) Core(TM) i5-8265U CPU. The run produces a probability distribution function (PDF) of the total energy after $50,000$ number of monte-carlo steps. By running `gnuplot plot.gnu` (this requires the software gnuplot) the user can compare the PDF obtained by their run with a PDF that we provide.



We acknowledge the support of the Swedish Research Council Grant No. 638-2013-9243 and 2016-05225. The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at PDC center for high performance computing.

References