Numerical Simulations for Vesicle Membranes Using
the Phase Field Model
Brady Thompson
Florida State University
Advisor: Xiaoqiang Wang

Abstract

In this paper, the phase field model is used to find a description for vesicle membranes with shapes and configurations not regularly found in nature. The phase field model is used to simulate the topological changes in the surface while conserving the principle that the elastic bending energy is minimized. The prescribed surface area remains unchanged as it is a constraint in the model. The method of how this experiment is done computationally is described using C++ code which seems to work well for the simulation. The method in which these results can be visualized using Matlab is also discussed in full detail. The results are often difficult to verify using biological experiments because of the irregularity of the initial shapes; however they are none the less projections of what can be studied in the lab and compare.

1) The Vesicle Membrane and Elastic Bending Energy

Lipid vesicle membranes are omnipresent in biological systems and are the building block to cellular life. The study of vesicle membrane shape transformation and assembly are very important for understanding the functionality of the cell. The cell membrane is composed of two basic components and form in one of three different patterns. The phospholipid bilayer of the cell membrane is made of molecules containing a hydrophilic head, hydrophilic meaning attracted to water, and also containing hydrophobic tails, hydrophobic meaning water avoidant. Because of these differences in the molecules they will spontaneously combine in these three ways, liposome, micelle, and bilayer sheet. (Figure 1.1) Each structure is very important in determining its applications. The goal of this study is to understand the deformation of vesicle membranes by any three variable equations through minimizing the elastic bending energy as described.

One reason the deformation of the cell membrane can be studied because of the cells permeability, or the ease in which molecules can pass through it. Knowing that molecules can pass easily through the cell membrane, it is known that the volume of a cell can fluctuate without surface area being changed.

There is one simplistic model in which the equilibrium shape of the cell membrane is largely determined by the curvature of the surface. The standard assumption of modeling the equilibrium shape of a cell is that as the elastic bending energy of the vesicle is minimized, simultaneously the equilibrium shape is discovered. The elastic bending energy is described by equation (1.1).
\[ E = \int_{\Gamma} a + b(H - c_o)^2 + cK \, ds, \] (1.1)

Where \( E \) is the elastic bending energy, \( \Gamma \) is the surface of the membrane, \( a \) is the surface tension, \( b \) is the bending rigidity, \( c \) is the stretching rigidity, \( H \) is the mean curvature which is described by \( H = \frac{k_1 + k_2}{2} \), where \( k_1 \) and \( k_2 \) are the principal curves that describe the membrane, \( K \) is the Gaussian curvature where \( K = k_1 k_2 \), \( c_o \) is the spontaneous curvature that describes the asymmetry effect of the membrane or its environment [1]. This equation (1.1) is accurate when describing a vesicle membrane that has a changing surface area and changing bulk volume. A much simpler equation may be used if the parameters are such that the surface area is a remained a constant. The first and last term of (1.1) may be neglected because they only matter with a changing surface area [1]

\[ E = \int_{\Gamma} \frac{k}{2} (H - c_o)^2 \, ds \] (1.2)

In some cases \( c_o \) may also be neglected if no spontaneous curvature is desired to be a part of the experiment. When the spontaneous curvature is zero this is referred to as an isotropic case, in which case the original equation (1.1) can be easily simplified to (1.3) and that is the equation we will use in our process [1].

\[ E = \int_{\Gamma} \frac{k}{2} H^2 \, ds \] (1.3)

One classical method for simulating and visualizing the minimization of elastic bending energy is to employ a mesh that has grid points on the interfaces. However, keeping track of large internal domain displacement may cause mesh entanglement and computational difficulties.
2) The Phase Field Model and the Equilibrium Shape

The phase field model is a very advantageous method in handling these complex changes in the topology. The phase field model used in this experiment can be viewed as a level set method with the goal to find and visualize the level set that symbolizes the surface of the vesicle. The phase field model in this project is introduced on a three dimensional computational domain Ω. This computational domain Ω is used to label the inside and the outside of the vesicle. The membrane Γ is set at the zero level of the phase field function. Besides being a level set function the phase field function has phases for different areas of the domain Ω. The phase field functions used to describe the membrane’s diffusive interface is:

\[
\varphi(x) = \tanh\left(\frac{d(x, \Gamma)}{\sqrt{2}\epsilon}\right)
\]  

(2.1a)

Where \(x\) is any point within the computational domain \(\Omega\). \(\varphi(x)\) has a maximum value of 1 and a minimum value of -1 and between there is a transitional layer with width measured by the parameter \(\epsilon\). The level set

\[
\{x : \varphi(x) = 0\}
\]  

(2.1b)

represents the surface of the membrane while the positive part represents the inside of the membrane, \(\{x : \varphi(x) > 0\}\), while the negative part represents the outside of the membrane, \(\{x : \varphi(x) < 0\}\). When the transition width \(\epsilon\) approaches zero, the phase field model with diffuse-interface becomes identical to a sharp-interface level set formulation (figure 2.1).

![Fig 2.1 Phase Field Illustrations](image)

Once the phase field function \(\varphi(x)\) has been established, in order to utilize it in combination with minimizing the elastic bending energy it needs be that there is an equation where \(E\) is a function of \(\varphi\). Through substitution and manipulation this combination becomes
\[ E(\varphi) = \frac{1}{\varepsilon} \int_{\Omega} (\varepsilon \Delta \varphi + \frac{1}{\varepsilon} (1 - \varphi)^2 \varphi)^2 dx \]  \hspace{1cm} (2.2)

Now that this equation is seen as the function we need to minimize the next step is to do exactly that, minimize. Basic calculus teaches that to find the minimum or maximum of a function, the derivative needs to be set to zero and solve for the max or min value. In our case, to find the derivative we will use the definition of the derivative that says:

\[ f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \]  \hspace{1cm} (2.3)

So, the derivative of (1.5) becomes:

\[ E'(\varphi) = \lim_{h \to 0} \frac{E(\varphi + h\psi) - E(\varphi)}{h} \]  \hspace{1cm} (2.4)

To start the minimization process we’ll first simplify

\[
E(\varphi + h\psi) = \frac{1}{\varepsilon} \int_{\Omega} (\varepsilon (\Delta \varphi + h\Delta \psi) + \frac{1}{\varepsilon} (1 - (\varphi + h\psi)^2 (\varphi + h\psi)) dx
\]

\[
= \frac{1}{\varepsilon} \int_{\Omega} \left( \varepsilon \Delta \varphi + \varepsilon h \Delta \psi + \frac{1}{\varepsilon} \varphi - 2 \varepsilon \varphi^2 - 4 \varepsilon \varphi h \psi + \frac{1}{\varepsilon} \varphi^3 + \frac{3}{\varepsilon} \varphi^2 h \psi + \frac{3}{\varepsilon} \varphi h^2 \psi^2 + \frac{1}{\varepsilon} h \psi - \frac{2}{\varepsilon} h^2 \psi^2
\]

\[ + \frac{1}{\varepsilon} h^3 \psi^3 \right)^2 dx \]  \hspace{1cm} (2.5)

Next we simplify \( E(\varphi) \) to make it simpler to combine with equation (1.10)

\[ E(\varphi) = \frac{1}{\varepsilon} \int_{\Omega} (\varepsilon \Delta \varphi + \frac{1}{\varepsilon} (1 - \varphi)^2 \varphi)^2 dx \]  \hspace{1cm} (2.6)

\[ E(\varphi) = \frac{1}{\varepsilon} \int_{\Omega} (\varepsilon \Delta \varphi + \frac{1}{\varepsilon} \varphi - 2 \varepsilon \varphi^2 + \frac{1}{\varepsilon} \varphi^3)^2 dx \]  \hspace{1cm} (2.7)

Now we combine (1.10) and (1.11) to get closer to finding our derivative

\[ E(\varphi + h\psi) - E(\varphi) \]

\[ = \frac{1}{\varepsilon} \int_{\Omega} \left( \varepsilon \Delta \varphi + \varepsilon h \Delta \psi + \frac{1}{\varepsilon} \varphi - 2 \varepsilon \varphi^2 - 4 \varepsilon \varphi h \psi + \frac{1}{\varepsilon} \varphi^3 + \frac{3}{\varepsilon} \varphi^2 h \psi + \frac{3}{\varepsilon} \varphi h^2 \psi^2
\]

\[ + \frac{1}{\varepsilon} h \psi - \frac{2}{\varepsilon} h^2 \psi^2 + \frac{1}{\varepsilon} h^3 \psi^3 \right)^2 - \left( \varepsilon \Delta \varphi + \frac{1}{\varepsilon} \varphi - 2 \varepsilon \varphi^2 + \frac{1}{\varepsilon} \varphi^3 \right)^2 dx \]  \hspace{1cm} (2.8)

Next we utilize the algebraic formula (1.13) to combine the two terms in (1.12)
\[ a^2 - b^2 = (a + b)(a - b) \]  

(2.10)

\[
E(\varphi + h\psi) - E(\varphi) \\
= \frac{1}{\epsilon} \int_\Omega \left( 2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi - \frac{4}{\epsilon} \varphi^2 - \frac{4}{\epsilon} \varphi h\psi - \frac{2}{\epsilon} h^2 \psi^2 + \frac{2}{\epsilon} \varphi^3 + \frac{3}{\epsilon} \varphi^2 h\psi \\
+ \frac{3}{\epsilon} \varphi h^2 \psi^2 + \frac{1}{\epsilon} h^3 \psi^3 \right) (\epsilon h\Delta\psi - \frac{4}{\epsilon} \varphi h\psi + \frac{3}{\epsilon} \varphi^2 h\psi^2 + \frac{1}{\epsilon} h\psi - \frac{2}{\epsilon} h^2 \psi^2 + \frac{1}{\epsilon} h^3 \psi^3)
\]

(2.11)

Next we incorporate the part of the limit in to the function and the fact that the definition of the derivative requires that all equation (1.14) is over \( h \). Since all the terms in the second factor have \( h \)'s in them it is easy to see that all the terms will be reduced by one power of \( h \). They will cancel with the \( h \) that is on the bottom of the fraction that was introduced by the definition of the derivative.

\[
E'(\varphi) = \lim_{h \to 0} \frac{E(\varphi + h\psi) - E(\varphi)}{h} 
\]

(2.4)

After the \( h \) on bottom is canceled out and all of the other terms with a power of \( h \) in them go to zero because of the limit we are left with

\[
E'(\varphi) = \frac{1}{\epsilon} \int_\Omega \left( 2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi - \frac{4}{\epsilon} \varphi^2 + \frac{2}{\epsilon} \varphi^3 \right) \left( \epsilon\Delta\psi - \frac{4}{\epsilon} \varphi \psi + \frac{3}{\epsilon} \varphi^2 \psi^2 + \frac{1}{\epsilon} \psi \right) dx
\]

(2.12)

Simplify

\[
E'(\varphi) = \frac{1}{\epsilon} \int_\Omega \left( \epsilon\Delta\psi \right) \left( 2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi - \frac{4}{\epsilon} \varphi^2 + \frac{2}{\epsilon} \varphi^3 \right) + \left( -\frac{4}{\epsilon} \varphi \psi + \frac{3}{\epsilon} \varphi^2 \psi + \frac{1}{\epsilon} \psi \right) (2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi \\
- \frac{4}{\epsilon} \varphi^2 + \frac{2}{\epsilon} \varphi^3) dx
\]

(2.13)

\[
E'(\varphi) = \frac{1}{\epsilon} \int_\Omega \psi \cdot \Delta \left( 2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi - \frac{4}{\epsilon} \varphi^2 + \frac{2}{\epsilon} \varphi^3 \right) + \psi \cdot \left( -\frac{4}{\epsilon} \varphi + \frac{3}{\epsilon} \varphi^2 + \frac{1}{\epsilon} \right) (2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi \\
- \frac{4}{\epsilon} \varphi^2 + \frac{2}{\epsilon} \varphi^3) dx
\]

(2.14)

\[
E'(\varphi) = \frac{1}{\epsilon} \int_\Omega \Delta \left( 2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi - \frac{4}{\epsilon} \varphi^2 + \frac{2}{\epsilon} \varphi^3 \right) \left( -\frac{4}{\epsilon} \varphi + \frac{3}{\epsilon} \varphi^2 + \frac{1}{\epsilon} \right) (2\epsilon\Delta\varphi + \frac{2}{\epsilon} \varphi \\
+ \frac{2}{\epsilon} \varphi^3) \cdot \psi dx
\]

(2.15)

We can compare (1.18) to an equation that helps us to see what part of this equation we really need to deal with and which part we don’t need to worry about
Now it is easy to find $G(x)$ in (2.15) if we compare the two equations, (2.16) and (2.15). If we minimize $G(x)$ it makes the problem much easier, since the closer $G(x)$ gets to zero the closer $E'(x)$ get to zero as well. After simplification:

$$G(x) = \frac{1}{\epsilon} \left( 2 \left( \epsilon \Delta \varphi + \frac{1}{\epsilon} (1 - \varphi)^2 \varphi \right) \right) + \frac{1}{\epsilon^2} (3 \varphi - 1)(\varphi - 1) \left( 2 \left( \epsilon \Delta \varphi + \frac{1}{\epsilon} (1 - \varphi)^2 \varphi \right) \right)$$

When $h$ is small $E(\varphi + h\psi) \leq E(\varphi)$, which means $E(\varphi_{n+1}) < E(\varphi_n)$. So that is the algorithm that is used when minimizing the elastic bending energy. The algorithm is written in C++ and will do thousands of iterations of minimizing, depending on the function we define phi to be originally.

As you can see, this algorithm has a Laplace transform embedded within the equations. Such a transform can cause computational complexities and increase the duration of the program’s minimization time by several orders of magnitude. The Fast Fourier Transform is a C runtime library that can compute discrete Fourier transforms like the Laplace transform that is presented in this algorithm. The Fast Fourier Transform package used in computing this algorithm was FFTW which stands for Fastest Fourier Transform in the West. The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson. When using the FFTW package, the difference in speed is substantial. Since this algorithm will have to run thousands of times, the speed that FFTW brings is a practically a necessity in performing the computations of the Laplace transform.

3) Visualization Process

The program’s code enables one to easily change the initial shape of the surface they want minimized. In all of the equations above where G or E is a function of $\varphi$, and we must still remember that $\varphi$ is a function of $x$. Before the program runs, it is requisite that $\varphi$ is defined by some function to give it the initial shape before minimization. In this specific line in the code, it is simple to see how one could easily change the initial shape defined by any three dimensional equations with three variables. This arbitrary equation that is written in this line is inside a for loop, which is nested inside a for loop, which are both nested inside a for loop. Also as defined by the person who runs the program, can choose how many times each loop runs. These numbers essentially give you the dimensions of your computational domain. In my computations, I chose these loops to run 64 times each. Thus giving me a three dimensional computational domain $\Omega$ such where there are 262,144 numerical entries in a 64x64x64 cube. For maximum visualization options and analysis, the program ran this section of the code every 25 to 50
iterations of the algorithm, each time writing 262,144 numerical values to .m file. The 
minimization program can run thousands of iterations before breaking out of an infinite loop. 
With thousands of iterations, means hundreds of .m files that can be processed to create many 
images revealing the step by step deformation of the vesicle.

The next step in the visualization process is to import all these files into Matlab. Once all 
the files have been imported into Matlab it is now time to run them in a file that will turn those 
values into a visible isosurface. What a contour map is to two dimensions is what an isosurface is 
to three dimensions. A contour map is a two dimensional array of numbers that have a gradient 
flow. With each level set marked by a different numerical value, a line can be drawn connecting 
all the values of a given level set. So the particular level set can be visualized in the entire array 
of numbers. In our situation we used a three dimensional array of numbers with a gradient flow, 
so just a line isn’t descriptive enough, but a surface connecting the values of a certain level set is 
needed to visualize the values throughout the array, this surface is called an isosurface.

The Matlab code I wrote does all the operations to the data set at in one go, to speed up 
the process of visualization. First the 262,144 values in the data set are reshaped into a three 
dimensional array 64x64x64 values in size. As defined earlier in 2.1b, the level set that 
symbolizes the vesicle is where \( \phi(x) = 0 \), so in the second line of the code, we set the isosurface 
to connect all the zero values into a surface. This is the code that is used to turn the code into an 
image.

1. v=reshape(data, 64, 64, 64);
2. p=patch(isosurface(v, 0));
3. isonormals(v, p);
4. set(p, 'FaceColor', 'blue', 'EdgeColor', 'none');
5. daspect([1 1 1])
6. view(24,12);
7. grid on;
8. camlight
9. lighting phong;

In this code, the color of the surface and angle in which the surface is viewed can be 
easily changed to maximize the viewing quality of the image.

4) Results

The simulations I ran were more focused on a bilayer sheet of arbitrary initial shape 
rather than a more spherical shape. Essentially, what happened was I created an equation to 
define an original shape of the vesicle bilayer. This initial shape would may or may not yield 
drastic change in the shape due to the minimization of elastic bending energy. The results and 
final images can often be very unpredictable, so it is important that the program run until the 
change from image to image is very small.
The Cayley Cubic as minimizes and the edges start to become full circle. The sharp twists and turns of the original shape eventually becomes a much more gradual change in the gradient flow. The original surface is described by the equation

\[
\text{radius} = -5(x^2y + x^2z + y^2x + y^2z + z^2y + z^2x) + 2(xy + xz + yz) \quad (4.1)
\]

![Figure 4.1 The Cayley Cubic](image)

The Clebsch surface is similar to the Cayley surface even though they have very different equation that gives them their original projection. However, differences do exist between the two simulations. The initial surface is described by equation

\[
\text{radius} = 81(x^3 + y^3 + z^3) - 189(x^2y + x^2z + y^2x + y^2z + z^2y + z^2x) - 54xyz + 126(xy + xz + yz) - 9(x^2 + y^2 + z^2) - 9(x + y + z) + 1 \quad (4.2)
\]

![Figure 4.3 The Clebsch Surface](image)

The changes in the Dingdong surface are very clear. See figure (4.3). The sheet and the sphere like membrane combine to create one sheet and then gradually become flat. The equation that describes the initial projection is:

\[
\text{radius} = x^2 + u^2 - (1 - z)z^2 \quad (4.3)
\]

This next image I call the Simpson’s image. It’s an equation I wrote that was in attempt to appear like a hole in time-space and after simulations it looks more similar to the power plant in the television show The Simpsons. It originates similar to a cylinder but after the bending energy is minimized it becomes much more like the inside of a torus. And as the width of the tube
becomes smaller and smaller, the curvature of the sides become more and more curved until the mean curvature is as close to zero as possible. See figure (4.4) The equation that describes the original projection is:

\[
\text{radius} = \frac{1}{x^2+y^2} - z
\]  

(4.4)

Figure 4.3 The DingDong Surface combines to make a flat surface

Figure 4.4 The Simpson’s surface begins to look more like the inner section of a torus

A spheroid’s edges change from being a convex shape and minimize to become a concave shape. See figure (4.5). The equation to describe the original shape is:

\[
\text{radius} = \frac{(x^2+y^2)}{a^2} + \frac{z^2}{c^2}
\]  

(4.5)

Another initial shape that I wanted to research was one I call, because of its horn shape, the Trumpet. The topological changes are small but still very significant. The tiny hole that is in the original shape is blown up until the mean curvature around that spot is zero. This is an important observation this happens in some simulations but not all. The hole in the Simpsons (figure 4.4)
seems to get smaller while the hole in the Trumpet (figure 4.5) and Whitney’s Umbrella (figure 4.6) seem to get bigger. The equation that describes the original surface is:

\[ \text{radius} = x^2 \cos(z) + y^2 \sin(x) - z^2 \cos(y) \] (4.5)

Figure 4.4 The spheroid’s edges become more concave then the original convex

Figure 4.5 The Trumpet from two different angles, both illustrate the expanding hole in the surface.

This last simulation was perhaps the most interesting. The original surface is called Whitney’s Umbrella. The top and bottom pinch off to create a tunnel about the top and bottom with a perpendicular tunnel in the center of the two that gets more and more circular. This one is a very
good example of how unpredictable the image with minimum bending energy will look like. See figure (4.6) The equation that describes the original surface is:

\[ \text{radius} = x^2 - y^2z \]  

(4.6)

Figure 4.6 two views of Whitney’s Umbrella deformation

5) Conclusion and Future Work

The cell membrane is a very dynamic material that is optimal for computational simulation. The discovery and research of vesicle membranes are always beneficial to the advancement in the medical field. The phase field model is an excellent method in evaluating the deformation of a vesicle membrane and its capacity for handling complex topological changes as we’ve seen here illustrates is value. Visualization of the vesicle is quite an easy process if the Matlab code presented earlier is used. The ability to see the change happening over time helps to develop a conceptual understanding of the vesicle deformation. The results of this work helps mathematicians to understand the value of the phase field model and how it can be used in a wide variety of studies. This work also helps biophysicists and biologists understand how mathematical models and computational methods can deepen their understanding of the cell membrane in ways that prove extremely difficult experimentally.
References
