University of Warwick

MOAC MSc Mini-project

# A Model for Cellular Blebbing

*Author:*
Matthew Lougher
0700304

*Supervisor:*
Till Bretschneider

May 11, 2012

# Contents

## Abstract

Using an approach based on the Active Contour Model, a model has successfully been designed and implemented in MATLAB to simulate the formation of blebs originating from isolated points of the plasma membrane (PM) which become detached from the actin cortex within a cell. In the system, connections between points on both the PM and the cortex, as well as links bridging the gap between the two structures, are modelled as Hookean springs, and the active contour is modelled using finite distance approximations of the derivatives of the contours formed by the PM and the cortex. The cell is also modelled as having an internal pressure, driving the PM outwards, which in turn is anchored in place by the cortex, allowing the formation of blebs when the two separate. This model has been proven to replicate the results of experimental observations by comparison of peak speed and displacement profiles of the blebs. It has also demonstrated that there is increase in peak speed at negative curvatures. Plus, when including the stochastic nature of links between the PM and cortex breaking and reforming, it has been found that there is a higher rate of bleb formation in regions of negative curvature, in agreement with expectations.

# 1 Introduction

## 1.1 What is Cellular Blebbing?

Cellular motility has often been thought to be solely dependant on the crawling effect generated by polymerisation of the actin cytoskeleton (CSK) cortex within the cell, triggered by the chemical environment in which the cell resides [1]. This leads to a protrusion pushing the plasma membrane (PM) of the cell in the direction of the chemical trigger, while simultaneously myosin II contracts the actin cortex at the rear of the cell, allowing retraction of the trailing edge, and hence overall movement of the cell [2]. However, in recent years it has been observed that blebbing, a process previously only associated with apoptosis and cell death [3], is another method of motion utilised by certain cell types [4].

Blebbing differs from actin polymerisation protrusions in that the local expansion of the cell results from the PM detaching from the cortex at a point, and the internal pressure forcing the PM into a hemispherical blister, known as a bleb, causing further detachment of the PM from the cortex [5]. A new region of cortex then forms within the bleb, which then reattaches to the existing cortex at the edges of the blister, completing the protrusion. The old cortex is also still present for a time after the bleb has been formed, and is known as an actin scar. A time series of images demonstrating blebbing can be seen in Figure 1 [6].

Blebs can occur in a stochastic fashion, with no explicit dependance on external triggers [7], however they can also be directed via the contraction (and strengthening) of the cortex elsewhere within the cell [8], which increases the cellular pressure and can lead to either weakening of the cortex [9] or separation of the PM from the cortex [10] It was also recently observed in *Dictyostelium Discoideum* that blebs seem to preferentially occur in regions of the cell that are concave, i.e. they have negative curvature locally along the PM, specifically at the sides of traditional actin protrusions [6].

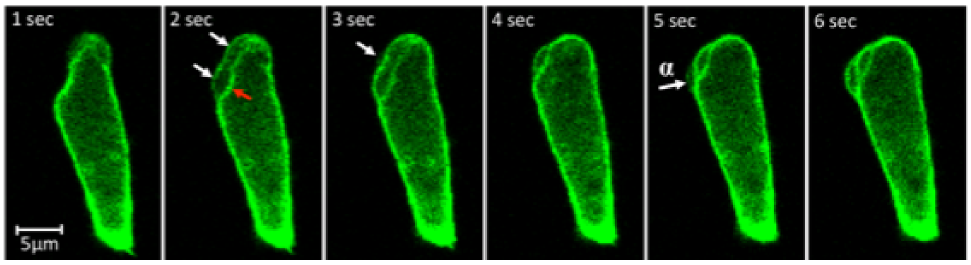
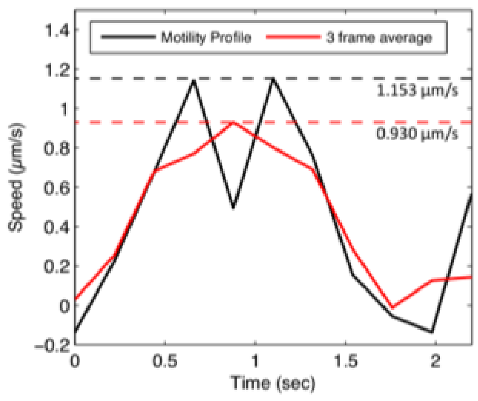


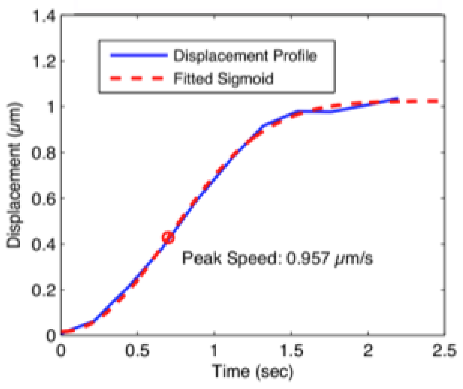
Figure 1: Time series of images illustrating the formation and evolution of blebs. The white arrows indicate bleb formation, while the red arrow indicates an actin scar.

## 1.2 Aims of the Model

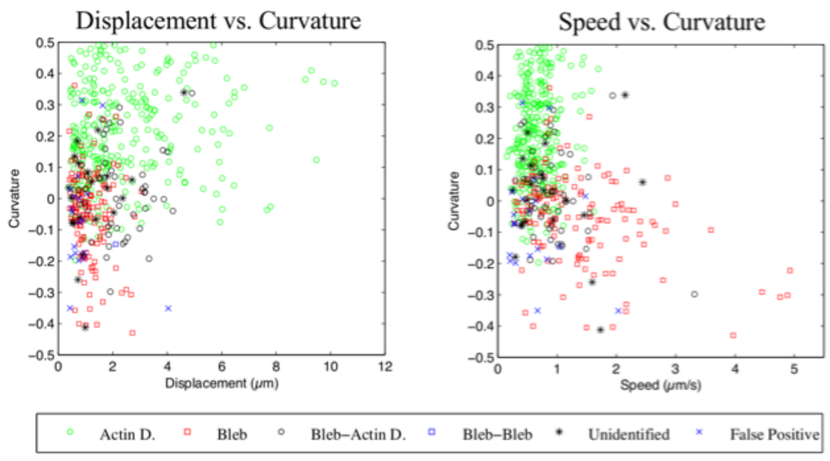
The aim of this research is to investigate the relationship between curvature and bleb formation by creating a mathematical model of a simple vesicle containing just an actin cortex, a PM and physical 'linkers' between the two. These linkers represent FERM domain proteins, which are a group of proteins that bind to the PM [11], some of which, e.g. filopodin, also bind to the actin cortex [12]. The focus of the model will be on bleb formation, or more explicitly, the separation of cortex and PM. It will not look at the recovery of the cell as there are added complications when trying to model the formation of new cortex within the bleb, and its subsequent re-connection to the rest of the cortex. The model will initially be required to simulate simple blebbing if a single linker is removed (representative of local weakening of the actin cortex) and will then be expanded to look at the effect of curvature, with the aim of replicating the previously mentioned observation.



(a) Bleb Motility Profile



(b) Bleb Displacement Profile



(c) Scatter Plots of curvature and maximum displacement/peak speed.

Figure 2: Experimentally obtained profiles of (a) speed and (b) displacement of a bleb as it grows, with different estimates of peak speed. (c) Illustrates how maximum displacement doesn't appear to be related to curvature, but how peak speed appears to increase with large negative curvature. Found in [6] as Figures 6a, 6c and 12 respectively.

Furthermore, analysis of the results from the simulations will be undertaken in a manner matching that done in [6] to compare the motility and displacement profiles of the bleb. It can be seen in Figure 2 that there are different methods of obtaining the peak speed at which the bleb grows, either from the peak of the raw data, the peak of the averaged data or by fitting a sigmoidal curve to the displacement profile (which attempts to compensate for noise in the data). As the model created in this work uses arbitrary units for distance it will not be possible to directly compare peak speeds with these results. However, the form of the profiles will be comparable, and the model will be modifiable in such a way that these profiles will be obtainable for a range of different curvatures, allowing the relationship between peak speed, as well as maximum displacement, and curvature to be investigated. It is expected that such an investigation will lead to a negative linear relationship between peak speed and curvature, and no distinct relationship between maximum displacement and curvature, as implied by Figure 2c. The basis of this expectation lies in the fact that in areas of negative curvature, the forces associated with the relaxation of the membrane, as well as the pressure from the cytoplasm (CYT) both act in an outward direction, encouraging linkers to break, as illustrated in Figure 3.

Another purpose of this investigation is to look into the stochastic nature of linker breakage and formation, and what effect this has on the number of blebs observed at positive curvature (which, if the above hypothesis is proven, would lead to slower blebs and so a greater chance of recovery) when compared to the number of blebs observed at negative curvature.

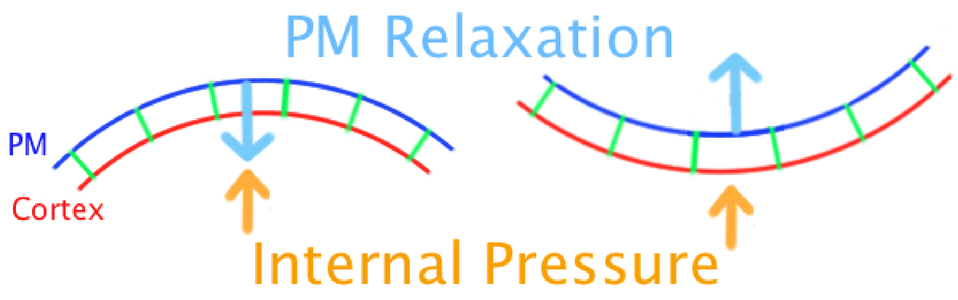


Figure 3: Schematic of forces acting on the PM in the case of positive (left) and negative (right) curvature. It can be seen that the linkers are under less stress when the curvature is positive as the pressure and PM relaxation forces oppose each other. For regions of negative curvature, however, both the pressure and PM relaxation forces sum together to pull the PM away from the cortex, putting increased strain on the linkers.

# 2 Materials and Methods

The model created for this research was coded using MATLAB R2011b, and the code can be seen in Appendix B.

## 2.1 Model Componenets

### 2.1.1 Active Contours

The basis of the model created was to treat both the actin cortex and the PM as active contours. These are traditionally used in computer imaging as a method of object detection, where the total energy, comprising the internal energy of the contour (or snake) and energy due to the image forces and other constraints, is defined as follows [13]:

$$E_{snake} = \int_0^1 \left(E_{internal} + E_{external}\right) \mathrm{d}s \tag{1}$$

This total energy is then minimised such that the internal forces (which cause the snake to shrink) balance the external forces (which provide boundaries around which the snake fits).

The internal energy can be defined as being dependant on the tension and curvature at a vertex $v$, where tension is proportional to the square of the first order derivative of the spread of points, and the curvature is proportional to the square of the second order derivative as follows:

$$E_{internal} = \frac{1}{2}\left(\alpha\left|\frac{\mathrm{d}v}{\mathrm{d}s}(s)\right|^2 + \beta\left|\frac{\mathrm{d}^2v}{\mathrm{d}s^2}(s)\right|^2\right) \tag{2}$$

Using computational methods to minimise $E_{snake}$ requires that the gradient of $E_{internal}$ be evaluated, which can be written as follows:

$$\nabla E_{internal} = \alpha\frac{\partial^2 v}{\partial s^2} + \beta\frac{\partial^4 v}{\partial s^4} \tag{3}$$

These derivatives can then be approximated using finite difference methods, utilising the values of nearest (second order) and next nearest (fourth order) neighbours. At this point, the assumption was made that all points defined by the contours (hereafter known as nodes) would be approximately equidistant for the most part, and that the separation between nodes to be used in the finite difference approximation, $ds$ should be defined as

$$ds = \frac{1}{N} \tag{4}$$

where $N$ is the number of nodes described by the contour. This argument is validated by the fact that the contour is already normalised by the integration in (1) being over the range 0 to 1.

Subsequently, the finite difference approximations required for (3) can be written as follows:

$$\begin{aligned}\frac{\partial^2 x}{\partial s^2} &\approx \frac{x_{s-1} - 2x_s + x_{s+1}}{(ds)^2} \\ \frac{\partial^4 x}{\partial s^4} &\approx \frac{x_{s-2} - 4x_{s-1} + 6x_s - 4x_{s+1} + x_{s+2}}{(ds)^4}\end{aligned} \tag{5}$$

where subscripts refer to the nearest and next nearest neighbours of the point $x_s$. (NB: $x$ simply refers to the cartesian x coordinate of the 2D vector $v$, and hence the above equations must be repeated for $y$.)

In order to reduce the computational power required to calculate these differentials, additional vectors were created such that neighbouring elements of $x$ and $y$ could be accessed easily, enabling vector calculations to be carried out rather than using iterative loops. These vectors (simply named $l$ and $r$) were then applied to themselves to allow access to the next nearest neighbours (creating vectors $ll$ and $rr$). Hence, (5) can be rewritten as:

$$\begin{aligned}\frac{\partial^2 x}{\partial s^2} &\approx \frac{x(l) - x}{(ds)^2} + \frac{x(r) - x}{(ds)^2} \\ \frac{\partial^4 x}{\partial s^4} &\approx \frac{x(ll) - x(l)}{(ds)^4} + \frac{x(rr) - x(r)}{(ds)^4} - \left(\frac{3}{(ds)^2} \times \frac{\partial^2 x}{\partial s^2}\right)\end{aligned} \tag{6}$$

The implementation of this can be seen in Appendix B.7.

### 2.1.2 Elastic Energy

In addition to the internal energy of the contour itself, this model will also incorporate the relaxation energy of the PM and cortex themselves as further internal energy terms. These will be included as elastic potential energy, with the PM, cortex and linkers all modelled as Hookean springs [14, 15], leading to the model illustrated in Figure 4.

Following Hooke's Law, the energy stored in each spring is defined as [16]:

$$E_{spring} = \frac{1}{2} k \left(L - l\right)^2 \tag{7}$$

where $k$ is the spring constant, $L$ is the resting length, and $l$ is the length of when compressed or stretched. As the model utilises the change in energy, (7) can be altered according to the relationship $Force = \frac{\mathrm{d}}{\mathrm{d}x} Energy$ to give the force acting on each node, which is the more familiar form of Hooke's Law [16]:

$$F_{spring} = k \left(L - l\right) \tag{8}$$

Of course, this force needs to be applied in the correct direction in order to minimise the energy in (7), and this is dealt with in section 2.1.5. Also, it should be noted that each node is connected to three springs, so (8) is a simplified definition of the force each node experiences due to the elastic forces.

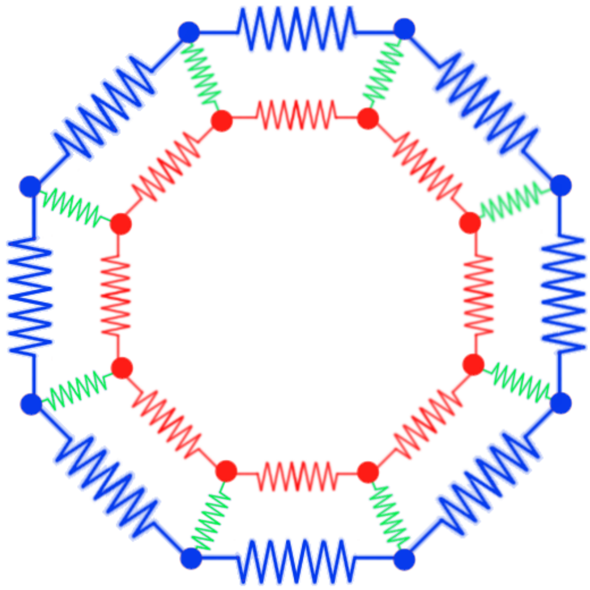


Figure 4: Schematic of how the PM (blue), cortex (red) and linkers (green) are modelled as Hookean springs. The solid circles denote the nodes of the PM and cortex. This simplified illustration therefore represents a system with just 8 nodes on each.

### 2.1.3 Pressure

A further addition to the internal energy in (1) is a term representing the inter-cellular pressure preventing the PM from collapsing. As shown in the previous section, the change in energy required to minimise $E_{snake}$ can be obtained using forces, which makes the pressure term considerably easier to deal with, simply using the relation $Pressure = Force \times Area$, where the area is calculated as the sum of all of the inter-node distances around the PM, allowing the force experienced by each node to decrease as the cell inflates. It should be noted that the cortex is mostly permeable to the cytoplasm (CYT) within the cell, and so this pressure term does not act on the nodes of the cortex [7].

### 2.1.4 Linker Interactions

As described in section 2.1.2, the linkers are modelled as springs connecting the nodes on the PM to the ones on the cortex. However, for the purposes of this model, the linkers will exert force on the PM nodes only. This is to balance the opposing force provided by the pressure term, and also to model the assumption that the cortex is held in place by forces much stronger than the linkers can exert, and as such is more rigid than the PM.

An additional consideration with regards to the linkers is that they are modelled as being fragile, and will break if the force exerted on them (directly proportional to the extension they experience) becomes too large. In this way blebs can propagate from an initial breakage, as the pressure forces the un-linked node outwards, in turn breaking the links between adjacent nodes and the cortex.

### 2.1.5 Directionality

All forces that act normally to the surface of either the PM or cortex are modelled as acting along the line perpendicular to the approximated tangent, as demonstrated in Figure 5. The implementation for this can be seen in B.6. The only exception to this is the force exerted on the PM by the linkers, which acts along the length of the linker itself.

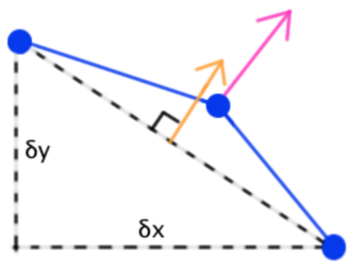


Figure 5: Exaggerated example of method used to approximate the local normal to the surface at each node. The pink arrow represents the“true” normal, while the orange arrow indicates the normal when approximated as being perpendicular to the line joining the neighbouring nodes.

One further stipulation that needs to be made is whether outwards or inwards forces are classed as positive or negative. It can be seen in the implementation code (Appendix B.5) that each differential equation has a minus sign in front of it, defining the direction needed to minimise the energy. From this start point, it then follows that outwards forces (e.g. pressure) must be negative, while inward forces (such as the tension on the PM due to the linkers) must be positive.

## 2.2 Model Parameters

As can be inferred from the previous sections, model parameters are $\alpha$, $\beta$, maximum linker length, force behind the isotropic pressure of the CYT and both elastic coefficients and natural lengths for each of the cortex, PM and linkers.

Initially, only $\alpha$ and $\beta$ were introduced, with values of $-10^{-7}$ and $10^{-11}$ assigned respectively. This resulted in a stable contour that neither shrunk down infinitely, as would be the case if $\beta$ were too small, nor bent itself into a shape with sharp corners, as would be the case if $\beta$ were too large. Next, the elasticity terms were included, with the natural length between nodes in both the PM and cortex set to arbitrarily be 1, such that there was no additional force pulling the contour closed (as would be the natural case if the distance between nodes in the cortex relaxed to be smaller than that between nodes of the PM), with the natural linker length set to be half this distance. The decision behind this value was that it made the resulting plots visually easy to interpret, as significantly larger linker lengths led to the cortex and PM being far apart, while shorter lengths led to them being indistinguishable on the plot.

Adding in the elastic terms also required values of the three elasticity coefficients to be determined, such that the system relaxed realistically. A value for $k_{PM} \sim 10^{-4}$ Nm$^{-1}$ was obtained from the supplementary information of [7], lending realistic values to the model. It was then assumed that the cortex and linkers should be modelled with an elastic coeffiecient an order of magnitude larger than that of the PM, such that ten times as much energy would be required to deform either of these by a comparable extension to the PM, hence fulfilling the requirement that the PM be more flexible than the other two components.

Finally, a pressure force was found by trial and error that led to stabilisation, and once the initial model had been set up and tested, it was found that setting the maximum linker length to just over double the natural linker length led to moderate bleb sizes.

## 2.3 Model Implementation

### 2.3.1 Shape

Initially, experiments were carried out on a circular vesicle, created by distributing the desired number of nodes between the limits of 0 and $2\pi$ and selecting an arbitrary radius for the PM, as well as a slightly smaller radius for the cortex to avoid zero values resulting from initial overlapping. This then allowed the allocation of cartesian coordinates, as can be seen in Appendix B.3. For some later experiments, this circle was modified by adding an extra angular dependance at a lower frequency, as can be seen in Appendix B.4, to create a shape with varying curvature.

### 2.3.2 Defining Node Separation

It had to be determined how to apply the opposing forces arising from the elasticity term for each node, as each would experience restoring forces from the connections on both sides. It was realised that calculating a simple average of the two forces effectively assigned each node with a separation value, $s$, describing the distance between the midpoints of the connections on either side, putting the node at the centre of the separation as follows:

$$\begin{aligned} s &= \frac{1}{2}\left[(s(r) - s) + (s - s(l))\right] \\ &= \frac{1}{2}\left(s(r) - s(l)\right) \end{aligned} \tag{9}$$

where $r$ and $l$ are the vectors described in section 2.1.1 utilised in vector operations. This value $s$ was then used in the calculation of the elastic force, replacing the $l$ in (8).

### 2.3.3 Loop Prevention

It was noticed that, in some instances, blebs formed with the PM displaying a tendency to loop around itself, as illustrated in Figure 6. It was initially thought changing the value of $\alpha$ would rectify this, but that led to destabilisation of the cortex. Instead, an additional stiffness term was included, representing the physical dimensions of the PM and cortex, such that the molecules themselves would exert an additional restoring force if bent beyond a certain limit, set to $90^{\circ}$. The implementation for this is shown in Appendix B.8,

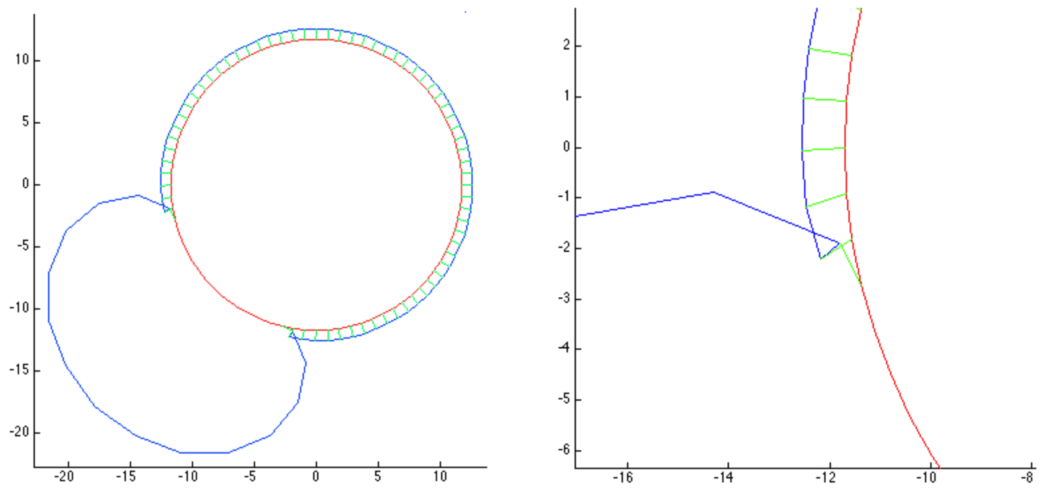


Figure 6: An example of a bleb whose formation has caused the PM to loop around itself. The red circle is the actin cortex, the blue contour is the PM and the green lines represent the linkers that have not been stretched to breaking point.

although it was found that this addition was not needed if the cortex was fixed, as the stability issue was not a problem, and so $\alpha$ and $\beta$ could just be adjusted in that case.

### 2.3.4 Solving the Differential Equations

In order to solve the differential equations rapidly and accurately, it was decided that MATLABs inbuilt ode45 function would be used, as it follows an adaptive time step method and utilises the Runge-Kutta (4,5) algorithm [18]. This algorithm is widely seen to be efficient for solving non-stiff initial value problems [19].

The ode45 function requires the input of a single function detailing the differential equation, with all variables contained in one column vector. This meant that any variables created within the function were not accessible via the workspace, and also that the system was manipulated (e.g. cutting linkers) by adding in *if* statements within the inputted function. The implementation of this can be seen in Appendix B.5.

### 2.3.5 Adding Finite Linker Width

In one experiment, it was decided that a minimum separation between the linkers should be implemented, representing the physical width of the linkers. In this way, regions with high linker density would experience an additional interaction due to the linkers competing for space, resulting in breakages. This was implemented first by simply cutting all linkers if their node on the PM had a corresponding separation $s$ (see (9)) which was less than a certain value to be determined by experiment.

The model was later modified to only cut linkers that were completely surrounded, therefore allowing ones whose neighbours had been cut to remain intact by effectively filling the vacated space. In order to do this as a vector operation, rather than using loops, vectors were created corresponding to the odd and even nodes on the PM. A random number was then generated to decide whether the even or odd linkers would be analysed first, and linkers were broken on all nodes in the selection that met the separation criteria described above. A NAND logical operation was then used on the linkers that had not yet been analysed, together with the breaking criteria, such that only linkers whose neighbours were still intact were themselves checked to see if their separation was less than the allowed minimum. The implementation of this can be seen in Appendix B.9.

### 2.3.6 Distributing Linkers Evenly

In order to look at the effect of curvature without the influence of increased linker density, an optional additional force acting at a point on the cortex was included, modelled as an external force according to (1). When implemented using the circular initial conditions, it was noted that as the vesicle was isotropic, any inward-acting force would have the same effect no matter where it was applied. In order to keep the model and coding simple, it was therefore decided that the deforming force would act solely at node 21, whose positional $x$ component is zero, hence the applied force would just act in the negative $y$ direction. The force was modelled as an additional elastic force, pulling the point on the cortex to a position closer to the centre of the cell, also with a zero $x$ component. The linkers were also not allowed to break until the cell had equilibrated in its new shape.

### 2.3.7 Stochastic Linker Breakages

When adding in a probability of each linker either breaking or reforming on their own, random numbers were generated for each node at each time step. Those greater than 0.99 were assigned a logic value of 1, and those less than 0.5 assigned a value of 0. A similar logic vector was also created corresponding to which linkers were still intact, and an XOR operation was used to determine which intact linkers would break, and which broken linkers would reform, each with a 1% probability of happening. Linkers exceeding the maximum length, however, remained broken, meaning that once a linker broke, there was only a certain amount of time for it to reform before the internal pressure forced the node past the maximum linker distance, at which point a bleb could form.

## 3 Results

For all experiments described below, the total simulation time was 50 s and the number of nodes was set to 80. In all cases the system was allowed to reach a stable equilibrium, before the property being investigated was altered at $t$ = 5.00 s. Also, the maximum linker length was set to 1.1, after which the linker would break.

Movies of the simulations shown in Figures 7, 9, 12, 13 and 16 can be found online at `http://www2.warwick.ac.uk/fac/sci/moac/people/students/2011/matthew_lougher/miniproj/blebmovies`

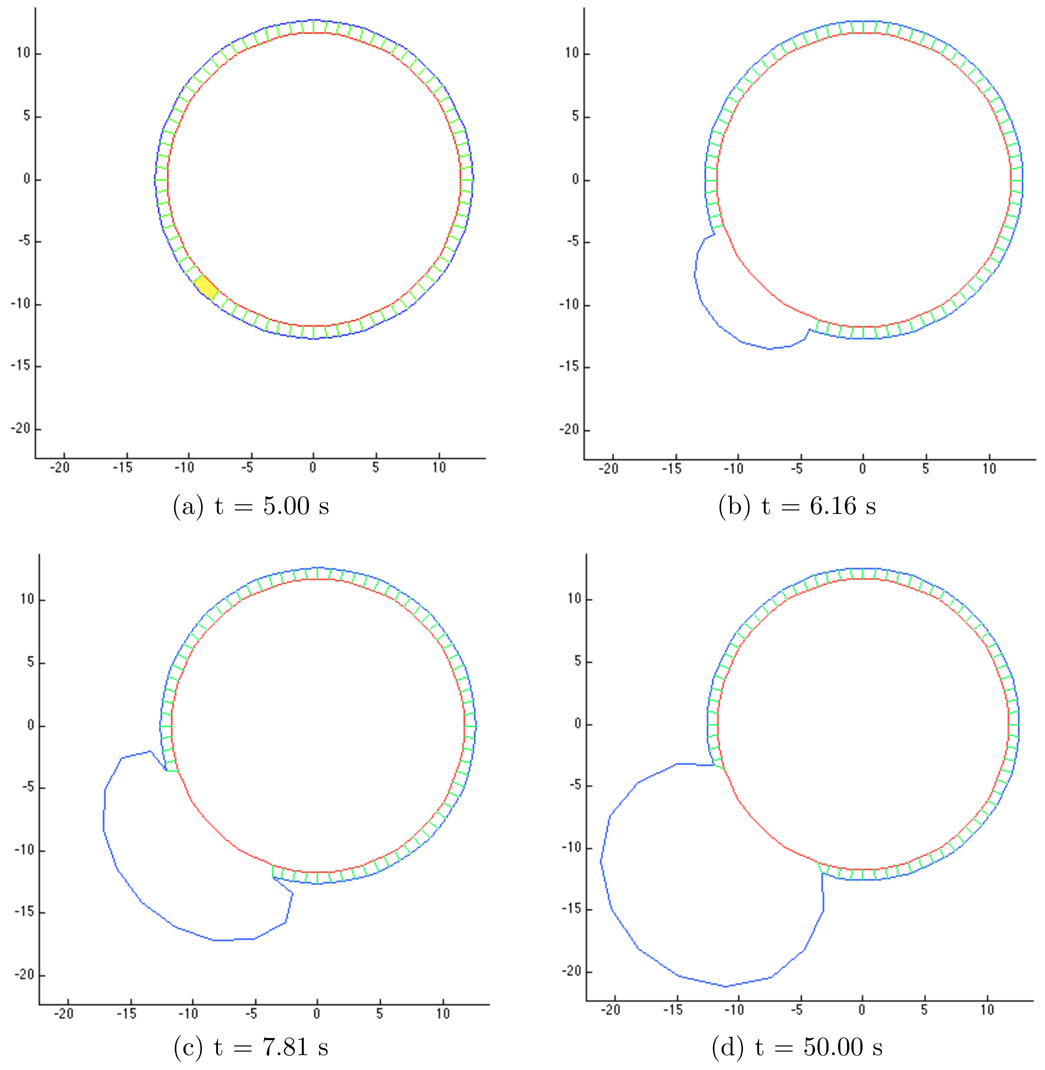


Figure 7: Illustration of bleb propagation after a single linker was cut at $t$ = 5.00 s. The area with the missing linker is shaded yellow in (a) for clarity.

## 3.1 Initial Blebbing & Pressure Dependence

As previously mentioned, the initial experiment on the model was performed with a circular vesicle. Figure 7 illustrates the bleb resulting from breaking a single linker at node 51[1]. The pressure force was set to 0.43 for this experiment after simple trial and error to get a value of the correct order of magnitude such that the simulation resulted in somewhere between none and all of the linkers breaking (found to be the case for a pressure force of 0.35 and 0.45 respectively).

As can be seen, once the linker is cut at $t = 5.00$ s the bleb quickly forms, with broken linkers forcing their neighbours to also break. This causes an unzipping effect until the PM has increased in surface area sufficiently that the pressure is no longer large enough to break any further linkers. In this case, this happened at $t = 6.16$ s, when the 10th and 11th linkers broke simultaneously (due to isotropic shape of vesicle). While not large enough to break any further linkers, the pressure then drove the PM further out, distorting it from the hemispherical shape initially, until it reached its maximum area at around $t = 7.81$ s. From this point on, the PM nodes that comprised the bleb then redistributed themselves evenly around the protrusion, again reforming a vaguely spherical shape as the simulation reached its end point at $t = 50$ s. This demonstrates bleb formation on a similar timescale to that observed in experiments of between 10 and 30 seconds [20], although the bleb formed is proportionally larger than observed.

Having proven the model to work, at least fundamentally, the role of the pressure exerted on the PM by the CYT was then investigated by gradually increasing the pressure force from 0.350 to 0.450, in increments of 0.001, and running the simulation at each pressure. The propagation of the bleb was quantified in each case by counting the number of nodes broken at the end of the simulation. The result of this experiment can be seen in Figure 8.

The results obtained were surprising in that there were no steady increases in bleb propagation as a function of pressure. Further investigation was done into the region around a pressure force value of 0.35, to see if there was indeed a steady increase in number of liners broken at higher resolution. It was found that in order to observe a variation from no additional links broken to just one, the difference in pressure force was of the order $10^{-15}$, and similarly for seven links

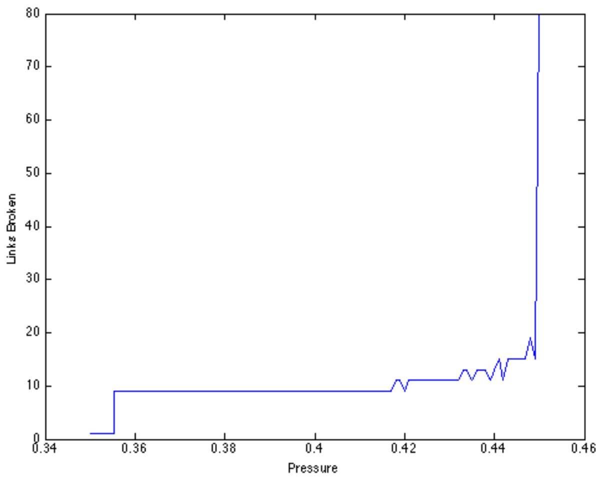


Figure 8: Plot of how the number of links broken varies with increased pressure force.

[1]Movie 1 online.

breaking to eight (on top of the initial one), although the jump from one to eight was still undetectable at this level. As such further investigation into this aspect of the blebs behaviour was ruled out.

## 3.2 Curvature with Varying Linker Density

When first investigating curvature, an alternate initial shape was created as described in section 2.3.1. However, it was noticed that this method placed nodes around the PM in such a way that linker density was increased in areas of negative curvature. As such, this was not a true test of the dependance on curvature, but it is included as an example of the effect of varying linker density. One thing that should be noted is that for this experiment the nodes of the cortex were fixed such that it didn't relax to a circular shape mid-experiment. Once again, a pressure of 0.43 was used, and linker 51 was cut after five seconds. Figure 9 shows the progression of the bleb expansion[2] .

It can immediately be seen that the way in which the bleb propagates is vastly different to the case with the circular vesicle. After the linker is cut, the PM detaches from the cortex evenly in both directions initially, until at $t = 6.19$ s when the protrusion reaches the region of negative curvature, and the high linker density there prevents the PM from detaching further. However, as the pressure force is still high enough to break linkers, the PM continues to detach in the other direction around the contour, until at $t = 9.75$ s when a total of 15 linkers have been broken. The bleb then expands in volume as before, until it stabilises in the shape in which it can still be seen after the full 50 seconds of the simulation.

In order to investigate this observation further, another experiment was run in which, keeping all other parameters constant, the simulation was run 80 times; each time cutting a different linker. The extent of bleb protrusion after 50 seconds was again quantified by counting the number of linkers broken, and this was plotted against the local curvature defined as [21]

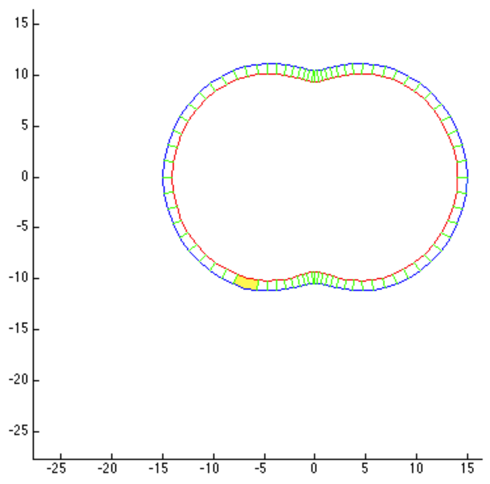
$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} \tag{10}$$

where $x' = \frac{\mathrm{d}x}{\mathrm{d}s}$ and $x'' = \frac{\mathrm{d}^2x}{\mathrm{d}s^2}$, and similay for $y$. As can be seen, in order to calculate this the finite difference method for the first order derivative was needed, which is defined below, along with that for the second order derivative defined in (6).

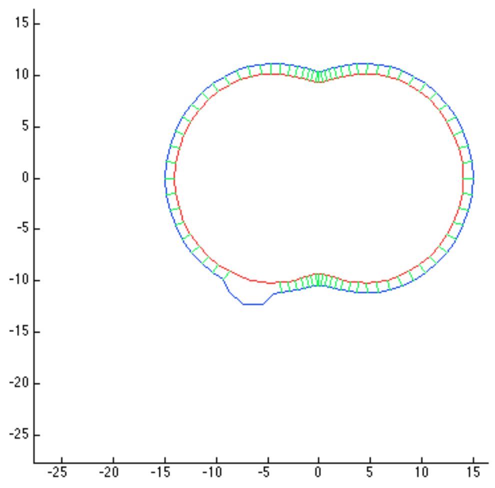
$$\frac{\mathrm{d}x}{\mathrm{d}s} \approx \frac{x(r) - x(l)}{ds} \tag{11}$$

The results from plotting the bleb propagation against this curvature can be seen in Figure 10a. As can be seen, the initial conclusion drawn is that, contrary to the

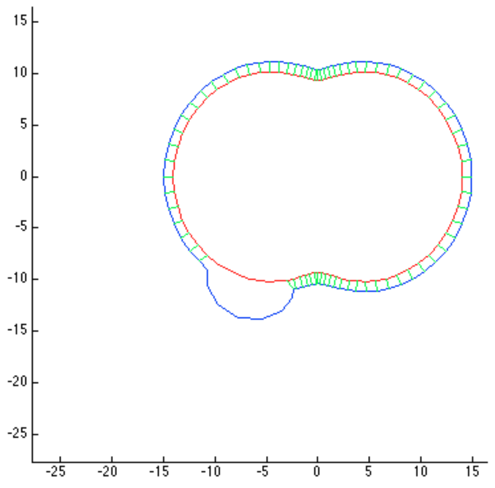
[2]Movie 2 online.



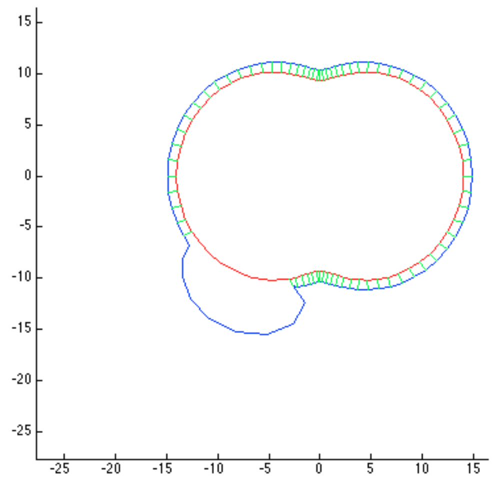
(a) t = 5.00 s



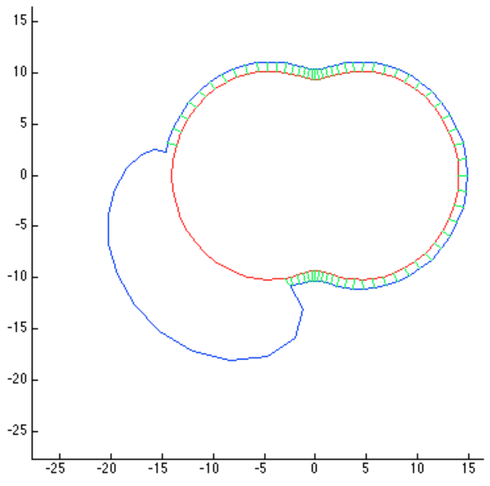
(b) t = 5.68 s



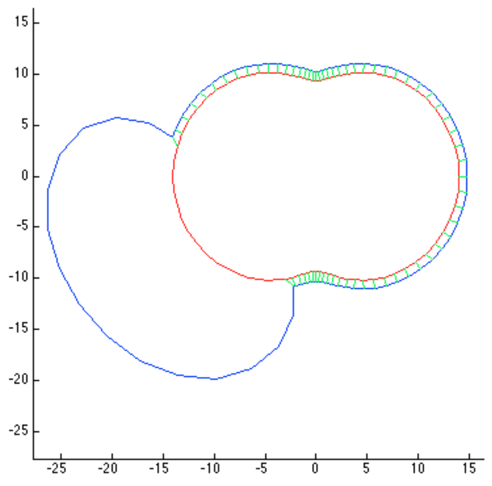
(c) t = 6.19 s



(d) t = 6.98 s

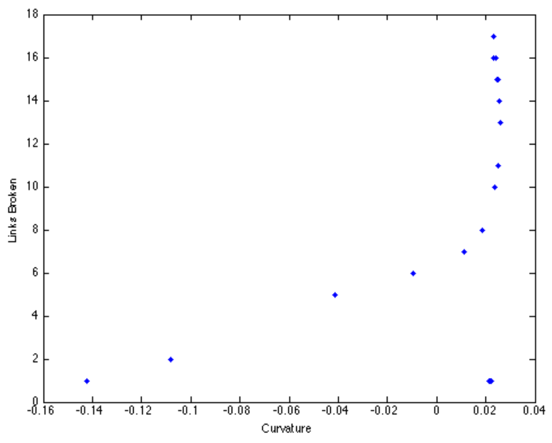


(e) t = 9.75 s

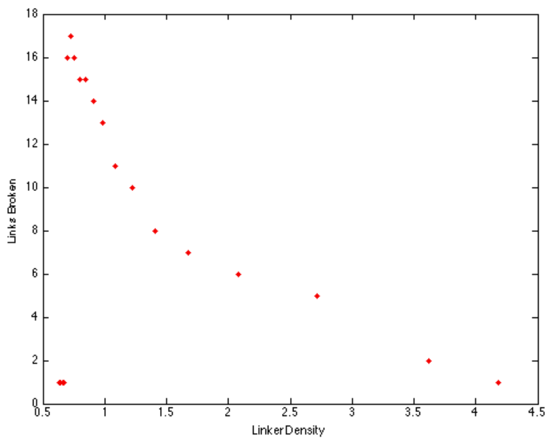


(f) t = 50.00 s

Figure 9: Illustration of bleb propagation in non-circular vesicle after a single link was cut at $t = 5.00$ s. The area with the missing linker is shaded yellow in (a) for clarity.



(a)



(b)

Figure 10: (a) shows a plot of the total number of linkers broken against the local curvature at the point where the initial linker was cut, and (b) shows the relationship between linkers broken and linker density.

initial hypothesis, blebbing is actually restricted by areas of negative curvature. However, Figure 10b shows that there is a slightly stronger relationship between linkers broken and linker density. As such, the exact nature of the relationships cannot be quantified by the model in this form. But, if it is in fact discovered that negative curvature does encourage blebbing, this experiment has shown that the effect of linker density dominates the curvature dependance, overriding the tendency to bleb.

## 3.3 Introducing Finite Linker Width

In an attempt to overcome the crowding of linkers into the negatively curved regions, it was thought that introducing a finite width for the linkers would force areas of high linker density to either spread out or form natural blebs.

As this experiment was run with the cortex fixed, the additional stiffness term was not needed and $\alpha$ and $\beta$ could be increased instead to prevent the looping effect mentioned in section 2.3.3. It was found that values of $\alpha = -5 \times 10^{-7}$ and $\beta = 5 \times 10^{-11}$ worked well.

Initial results cutting all linkers with separations less than a minimum distance are shown in Figure 11, demonstrating the number of linkers broken as the minimum distance was changed. As with the pressure dependance, there is a large jump in bleb

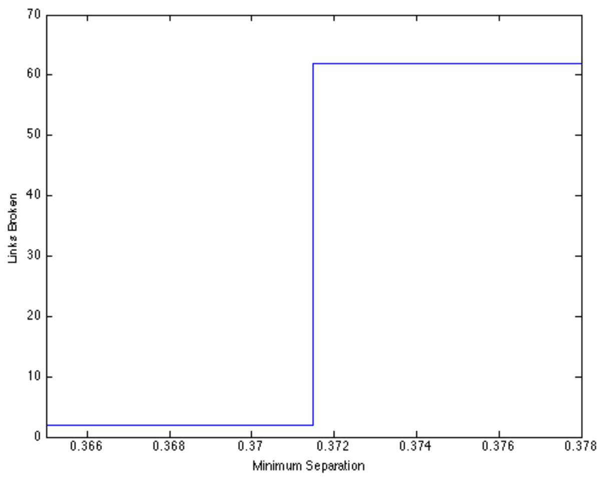


Figure 11: Plot of bleb propagation, quantified as the number of linkers broken, against the minimum linker separation.

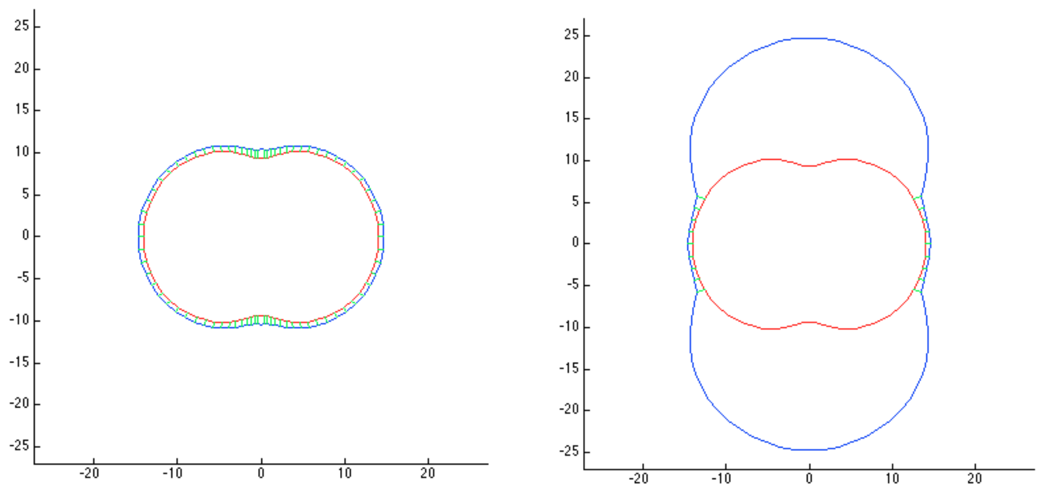


Figure 12: Plots of the final model configuration after running two simulations with linker width set to 0.387. These simulations were identical apart from the fact that one analysed the even numbered linkers first, while the other analysed the odd numbered ones first.

propagation apparent at a critical separation, which is still there at much higher resolution (changes in distance on the order of $10^{-7}$), although this time the jump was from only the centre-most linkers breaking in each negative curved region to over 75% of linkers breaking.

At this point it was decided to implement the further conditions to restrict the linkers that can break, as described in section 2.3.5. However, the random allocation part of the code leads to the results shown in Figure 12, indicating the massive change in bleb size between running near identical simulations[3]. Obviously, this deviation doesn't lead to reliable results, and so another approach was attempted.

## 3.4 Curvature with Fixed Linker Density

When performing this simulation, the cortex had to be allowed to change with time, so as to allow for the local distortion. However, adding in this extra term destabilised the cortex when using values of $\alpha$ and $\beta$ previously determined without the additional force (see section 2.2), and also led to complete detachment of the PM when using the values utilised in the experiment looking into finite linker width (see section 3.3). As such, it was discovered that in fact the model works best under the current conditions with different values of $\alpha$ and $\beta$ for the PM and cortex, indicating a further difference in properties of the two components, as well as the ten fold difference in elastic coefficients already implemented. The values used for this part of the experiment

| | Cortex | PM |
|---|---|---|
| $\alpha$ | $-5 \times 10^{-7}$ | $-1 \times 10^{-7}$ |
| $\beta$ | $5 \times 10^{-11}$ | $1 \times 10^{-11}$ |

Table 1: Values of $\alpha$ and $\beta$ for the cortex and PM used to change the shape of the vesicle while keeping the linker density constant.

[3]Movie 3 online.

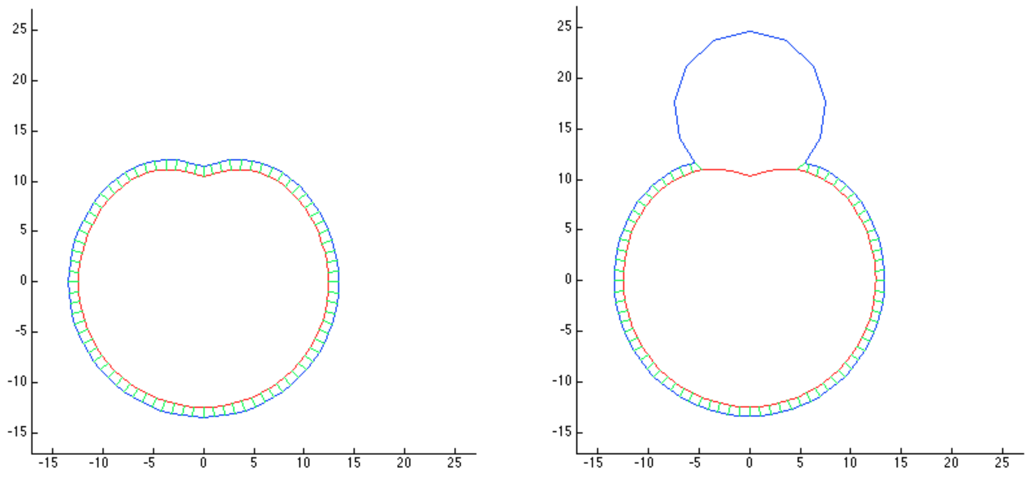

(a) t = 9.25 s (b) t = 50.00 s

Figure 13: Demonstration of bleb formation originating from point of negative curvature, with linker density kept uniform.

are shown in Table 1.

The results of adding in the additional distorting force[4] can be seen in Figure 13. While the additional force was first applied at $t = 1.00$ s, and the linkers allowed to break after $t = 5.00$ s, the bleb doesn't begin to form until over 4 seconds later, indicating that the force applied is only just enough to stretch the linkers to breaking point. What is observed, however, is that after the initial linker breaks, a bleb forms as expected around the area of local negative curvature. The bleb was formed from 9 nodes breaking, which is 2 less than the number that broke when a single linker was cut in an undistorted circular vesicle under the same pressure force of 0.43, as seen in Figure 8. The reason for this is likely to be due to the change in cortex shape, as, as can be seen in Figure 13b, the linkers at the edge of the bleb are attached to a relatively flat part of the cortex.

[4]Movie 4 online.

# 4 Discussion

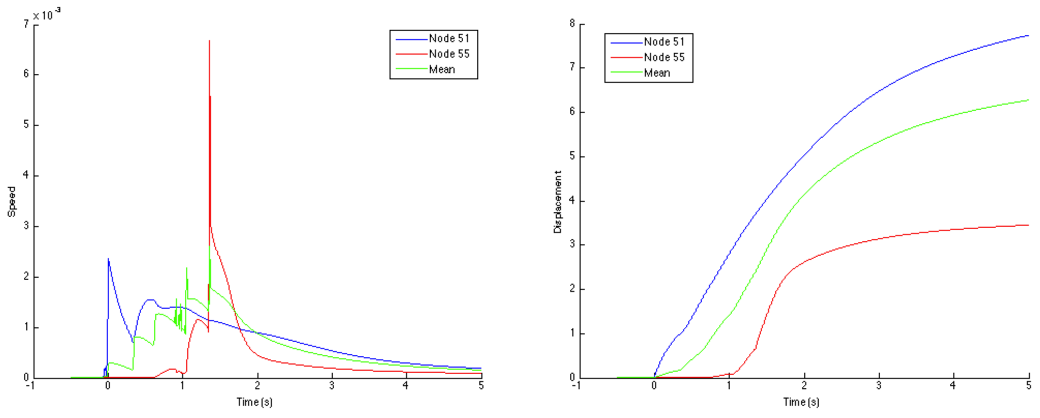
The following section describes analysis carried out on the matrices outputted from the model as a series of positions of the nodes over time.

## 4.1 Velocity and Displacement Profiles

In order to compare the simulation data to that displayed in Figure 2, velocity and displacement profiles were created using a simple *for* loop that performed vector operations on a selected node for each time step, then outputted either the immediate distance each node 'travelled' during each time step, or the cumulative distance the node had moved since bleb formation began.

Figure 14 illustrates the velocity and displacement profiles from the data obtained from the simulation shown in Figure 7, i.e. for a circular vesicle with a linker cut manually. The time displayed has been offset such that the bleb begins to form at $t = 0$ s. One thing to note is that due to the arbitrary nature of the distances used in the model, it is impossible to directly compare actual values at this stage. As such, the main comparisons drawn will be in the shapes of the profiles, to ensure the correct behaviour of the model.

It can be seen that there is a difference in the profile shapes for nodes 51 and 55, and a further difference between each of these and the mean across all nodes that form the bleb. As well as the expected lag in the velocity profiles that demonstrates the time taken for the bleb to progress from node to node, there are also spikes apparent, which correspond to the individual linkers being cut, and the instantaneous acceleration that the linker experiences at that time. As such, the mean profile will be used for comparison to the profiles in Figure 2.



(a) Velocity Profile (b) Displacement Profile

Figure 14: Velocity and Displacement profiles for nodes 51 and 55, along with the mean profile for all nodes that comprise the bleb. The simulation was run with an initially circle vesicle with linker 51 cut manually. The time axis is offset such that the bleb begins to form at $t = 0$ s.

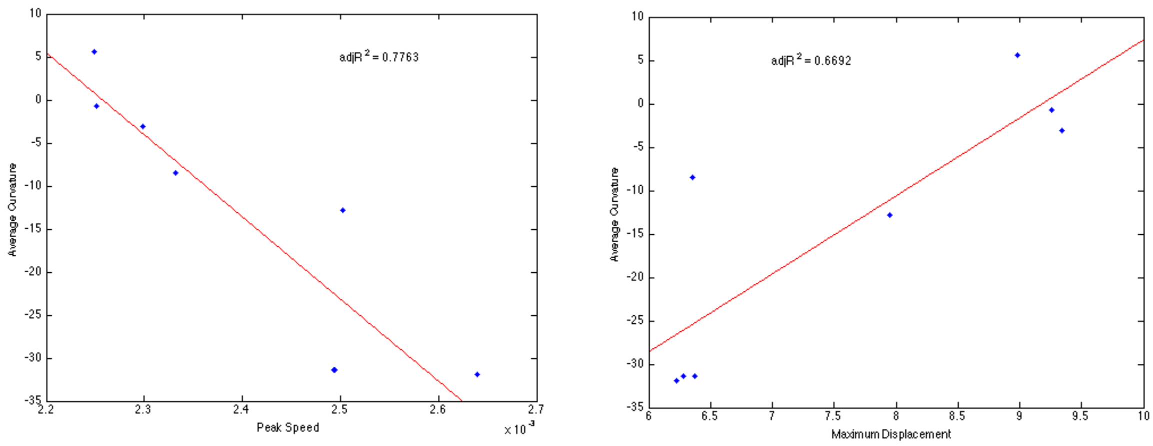
It can instantly be seen that the shape of the velocity profile is similar in both cases; although the massive acceleration in the simulated data representing the moment each linker breaks is not apparent in the observed data. It must also be considered that the model does not incorporate reformation of the cortex or motion of the rest of the cell around the bleb, and as such the apparent exponential decay in velocity in the modelled data after reaching the peak may not be a wholly accurate representation. Furthermore, it can be seen that the displacement profiles for the individual nodes, as well as the mean, seem to follow a sigmoidal pattern, as expected and described in [6].

It is believed that these profiles confirm that the model does indeed replicate bleb behaviour to an acceptable degree, and as such can now be used to look into the effect of curvature on bleb propagation.

## 4.2 Curvature

Utilising the above result that individual linkers may not be a true representation of the bleb behaviour, the following analysis again follows the average velocity and displacement of all nodes that form the bleb. The simulation was run as detailed in section 3.4, with node 21 pulled inwards to $y$ values between 8.5 and 10.5, hence giving different local curvature and allowing for different measures of bleb propagation. In this instance, however, the curvature of the bleb was calculated as the average curvature over the region which the bleb formed, rather than the curvature at the point where the bleb originates, so as to match the averaged velocities and displacements. The resulting scatter plots can be seen in Figure 15.

It is apparent that, while not as linear as was hoped, Figure 15a indicates that there is indeed a link between the curvature of the bleb site and the peak velocity reached by the bleb as it expands. This result is in agreeance with the observations of Figure 2,



(a) Peak Velocity Distribution (b) Maximum Displacement Distribution

Figure 15: Scatter plots of how the average peak velocity and maximum displacement across the bleb vary with the average curvature of the cortex at the bleb site.

although the deviations from a simple relationship indicates that the model is perhaps oversimplified for looking at this property. Figure 15b, however, indicates that there may indeed by a weakly positive relationship between curvature and average total displacement of the nodes within the bleb, contradicting the evidence in Figure 2. It must be considered, however, that these plots have been made with a very limited dataset determined by the parameters of the model, and so more experiments should be undertaken with a greater variation in initial conditions in order to examine the above conclusions.

## 4.3 Stochastic Linkers

In order to investigate the stochastic nature of linkers breaking and reforming naturally, an additional condition was imposed as described in section 2.3.7. The simulation was run multiple times until a total of 300 blebs had been observed, with the node at which each bleb originated being recorded. Figure 16 shows an example vesicle at the end of one of these simulations[5], with blebs originating from nodes 5, 21 and 54. Additional protrusions, that have not yet formed blebs, are identifiable at nodes 36, 66, 75 and 77. These isolated broken links have gone past the point at which they could be recovered, and so if the simulation were run for longer, of it the internal pressure of the cell had not been reduced by the existence of the other blebs, these points would have matured into full grown blebs,

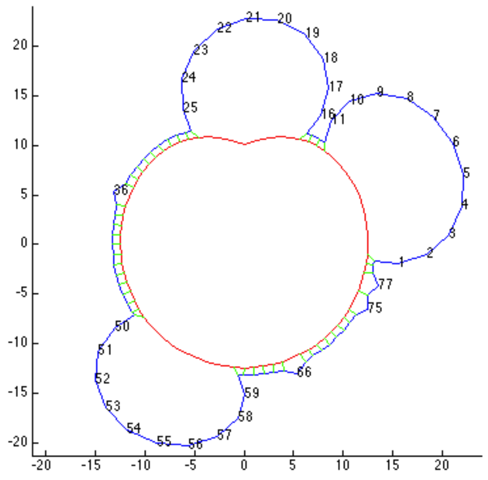


Figure 16: Example of blebs formed when linkers are given random chance of breaking or reforming.

[5]Movie 5 online.

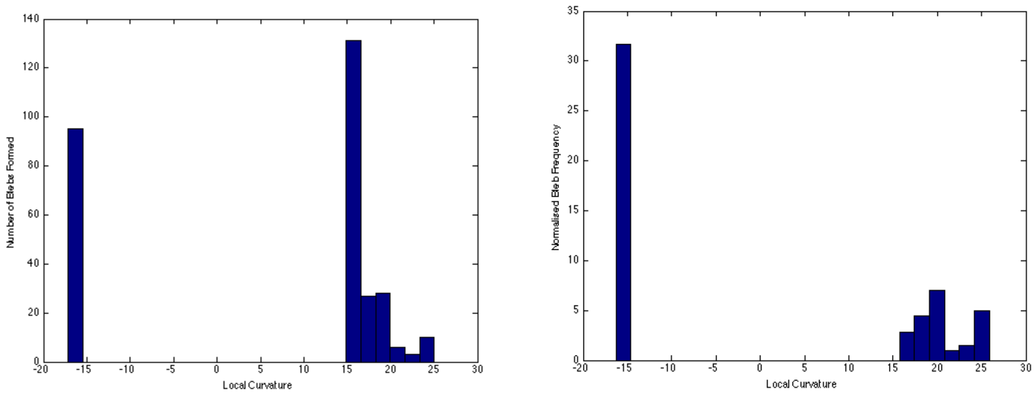


Figure 17: Histograms of the number of blebs formed against the local curvature at the site of formation. (a) shows the raw data, while (b) shows the data normalised against the number of nodes in the system with that curvature.

and as such were included in the data recorded as bleb sites.

Figure 17 shows the data collected into histograms, where the number of blebs is displayed against the average curvature at node from which the bleb originated. The average curvature was calculated using the curvatures of the node whose linker originally broke, along with its nearest and next-nearest neighbours, utilising 5 individual curvature values in all. The initial result from Figure 17a seems to indicate that while negative curvature does lead to a high rate of bleb formation, more blebs are formed at positive curvature. However, this does not take into account the amount of nodes that have that curvature, and so Figure 17b shows the same data set normalised against the incidence of each curvature value. In this instance it is clear to see that there are vastly more blebs formed per node in the region with negative curvature.

This result can justifiably be explained by the conclusion in section 4.2 that blebs propagate faster in regions of negative curvature, and as such take less time to move outside the distance in which stochastic reformation of the linkers is possible.

## 4.4 Linker Angle & Separation

While it was briefly investigated in section 3.3 that bleb formation may have a relationship to the separation between nodes on the PM, it was not discussed whether bleb formation might also be dependant on the angles between adjacent linkers. As can be seen in Figure 18, there is a strong relationship between the inter-linker angle and the nodal separation on the PM, and so it is conceivable that there is also a relationship between angle and bleb formation.

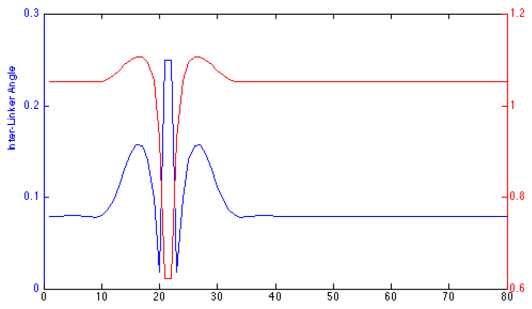


Figure 18: Plots of how the inter-linker angle and separation vary around the circular vesicle distorted by applying force to node 21.

However, there was not time to investigate this possibility further, and it remains an open question for further investigation.

## 4.5 Further Work

Although the work presented here has led to several conclusions, there is more work that can be done with this model. Firstly, the model itself needs refining in a way which became apparent when investigating the stochastic nature of bleb formation, as it was observed that occasionally two blebs would form that overlapped each other. While this was not of immediate concern at that point, for that particular experiment was concerned with the points from which the bleb originated, it is something that must be looked into if more work is carried out on systems in which multiple blebs can form. In order to do this, the code must be modified such that the vectors between nodes on the PM may not cross at all. Implementing this would possibly also remove the need for the additional stiffness term, as it should do the same job of ensuring that lines between nodes don't cross, especially if it is also extended to include the vectors between nodes on the cortex. However, this has the potential to be very computationally intensive, as it would require checking that the lines between each adjacent pair of nodes doesn't cross any other line in the whole system.

As far as further investigation using the model is concerned, there are still areas to be explored with regards to the effect curvature has on bleb formation. Namely, a system needs to be created that has a wider range of curvature values, such that a more concrete relationship without discontinuities may be formed. One possible solution may be to take a step away from the realism offered by the circular vesicle and to instead use a linear cortex fixed in the shape of a sin wave. If the ends of the PM were also fixed in position, the system would then provide a greater, smoothly varying range of curvature values. This system could then also look again at linkers that break stochastically, and link the results back to the case of a closed loop vesicle.

Also, the possibility of relationships between bleb propagation and both the inter-linker angle and linker separation ought to be investigated further, as described in section 4.4. In order to do this, a simple circular system could be used, therefore keeping curvature constant, with linkers spaced at varying distances around the vesicle, allowing direct investigation into the effects of separation and angle.

Another possible investigation would be to deform the vesicle with an outward force, simulating a traditional actin protrusion. This would then lead to negatively curved regions at the sides of the protrusion, from which it is theorised that blebs will preferably form. This was attempted with the model during this work, but resulted in destabilisation of the cortex. Further work would be needed to rectify this problem in the model and therefore investigate this effect.

A further addition to the model would be the inclusion of realistic lengths and sizes for the PM and cortex, as well as the FERM proteins that comprise the linkers. This would then allow quantitative comparison to experimental results, rather than simply qualitative.

It would also be interesting to look into why, in some instances, two blebs can form on the PM and be held in place, and thus prevented from fusing, by a single linker. This is not expected behaviour, as it would be reasonable to assume that the combined force exerted by these two blebs would be enough to break the final linker.

Finally, the model could be drastically increased in complexity to also simulate the recovery of blebs by formation of a new actin cortex beneath the PM. However, many more terms would likely be needed to accurately model this, therefore making the model both harder to create and more computationally expensive to run.

## 5 Conclusions

The aim of this research was to create a model which both replicated and expanded upon observed relationships between the propagation and rate of formation of blebs with the curvature of the cortex and PM at the point at which the bleb originated.

By comparing velocity and displacement profiles of data from the model with those from experiment, it has been shown that the model created does indeed replicate the observed activity.

Also, again by comparing velocity profiles, it has been shown that there does indeed appear that the peak speed of blebs increases with negative curvature. This is likely due to the summative effect of the relaxation forces of the PM working in the same direction as the force exerted by the pressure in the CYT to accelerate the PM away from the cortex faster, leading to higher speeds.

An unexpected possible relationship between curvature and maximum bleb displacement has also been observed. However, the relationship is not as concordant as that obtained from the velocity data, and so more simulations would need to be run with a greater variation in curvature in order to see if this effect is just coincidental.

It has been proven that even with linkers breaking and reforming in a stochastic fashion, as seen in experiment, that there is a preference for blebs to form in areas of negative curvature. This agrees with the above result of bleb speed being greater in these areas, as if the PM is moving away from the cortex at a greater speed, there is less likelihood of the linker randomly reforming before the PM exceeds the maximum linker length.

## 6 Acknowledgements

I would like to extend thanks to my supervisor for this work, Till Bretschneider, for his guidance throughout this project, as well as his patience while I grasped the finer points of my task. Thanks also go to Richard Tyson for offering help and advice when I needed additional guidance with interpreting some of the results of my model, and for help refining my MATLAB code at times.

I would also like to thank the MOAC DTC, as well as the EPSRC for allowing me the opportunity to carry out this work and providing both the software and hardware upon which the model was constructed.

## References

[1] S. M. Rafelski, J. A. Theriot, Crawling Toward a Unified Model of Cell Motility, *Annu Rev Biochem*, **73**, 209-239 (2004).

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology of the Cell, 5th Edition* (Garland Science, New York) **2008** pages 1036-1039.

[3] J. C. Mills, N. L. Stone, J. Erhardt, R. N. Pittman, Apoptotic Membrane Blebbin is Regulated by Myosin Light Chain Phosphorylation, *J Cell Biol*, **140**, 627-636 (1998).

[4] O. T. Fackler, R. Grosse, Cell Motility Through Plasma Membrane Blebbing, *J Cell Biol*, **181**, 879-884 (2008).

[5] G. Charras, E. Paluch, Blebs Lead the Way; How to Migrate Without Lamellipodia, *Nat Rev Mol Cell Biol*, **9**, 730-736 (2008).

[6] R. A. Tyson, *The Cartography of Cell Motion*, Thesis (PhD), University of Warwick (2011).

[7] J. Brugués, B. Maugis, J. Casademunt, P. Nassoy, F. Amblard, P. Sens, Dynamical Organization of the Cytoskeletal Cortex Probed by Microscopic Aspiration, *PNAS*, **107** 15415-15420 (2010).

[8] B. Maugis, J. Brugúes, P. Nassoy, N. Guillen, P. Sens, F. Amblard, Dynamic Instability of the Intracellular Pressure Drives Bleb-Based Motility, *J Cell Sci*, **123**, 3884-3892 (2010).

[9] E. Paulch, M. Piel, J. Prost, M. Bornens, C. Sykes, Cortical Actomyosin Breakage Triggers Shape Oscillations in Cells and Cell Fragments, *Biophys J*, **89**, 723-733 (2005).

[10] J. Dai, M. Sheetz, Membrane Tether Formation from Blebbing Cells, *Biophys J*, **77**, 3363-3370 (1999).

[11] A. Chishti *et. al.*, The FERM Domain: a Unique Module Involved in the Linkage of Cytoplasmic Proteins to the Membrane, *Trends Biochem Sci*, **23**, 281-282 (1998).

[12] G. Gerisch *et. al.*, Actin-Associated Proteins in Motility and Chemotaxis of Dictyostelium Cells, *Symp Soc Exper Biol*, **47**, 297-315 (1993).

[13] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active Contour Models, *Int J Comput Vision*, 321-331 (1988).

[14] W. Helfrich, Elastic Properties of Lipid Bilayers: Theory and Possible Experiments, *Z. Naturforsch*, **28**, 693-703 (1973).

[15] H. Coskun, Y. Li, M. Mackey, Ameboid Cell Motility: A Model and Inverse Problem, with an Application to Live Cell Imaging Data, *J Theo Biol*, **244**, 169-179, (2006).

[16] J. W. Jewett, R. A. Serway, *Physics for Scientists and Engineers with Modern Physics, 7th Edition* (Thompson learning, London) **2007** page 426.

[17] R. L. Burden, J. D. Faires, *Numerical Analysis, 8th Edition* (Brooks/Cole, Belmont) **2005** page 257.

[18] MATLAB support documentation - ode45, `http://www.mathworks.co.uk/help/techdoc/ref/ode45.html` (2012) - *retrieved 25th April 2012*

[19] J. Dormand, P. Prince, A Family of Embedded Runge-Kutta Formulae, *J Comput Appl Math*, **6**, 19-26 (1980).

[20] J. Tinevez, U. Schulze, G. Salbreux, J. Roensch, J. Joanny, E. Paluch, Role of Cortical Tension in Bleb Growth, *PNAS*, **106**, 18581-18586 (2009).

[21] F. Mokhtarian, A. Mackworth, A Theory of Multiscale, Curvature-Based Shape Representations for Planar Curves, *IEEE Trans Pattern Anal Mach Intell*, **14**, 789-905 (1992).

## A Abbreviations used

| | |
|---|---|
| CSK | Cytoskeleton |
| CYT | Cytoplasm |
| PM | Plasma Membrane |

# B MATLAB Code

The following subsections contain the MATLAB code for the model, separated into several different functions for ease of use. They are listed here in the order in which they are called.

Files can be downloaded online from http://www2.warwick.ac.uk/fac/sci/moac/people/students/2011/matthew_lougher/miniproj/activecontourmodel.zip.

## B.1 ActCont_ode.m

```
function [x_m,y_m,x_c,y_c,T,links] = ActCont_ode(draw)
% Creates active contour blebbing model.  Input 0 for no graphical output,
% 1 if want to plot progression of the system, or 2 for just final
% configuration.

% Retrieves simulation parameters, stored in external function.
P = Parameters_ode;

% Creates initial membrane & removes duplicated end point.
initial = CircularInitial(P.Nodes+1); % Creates circular membrane.
% initial = InfInitial(P.Nodes+1);      % Creates squashed membrane.
initial = initial(1:P.Nodes,:);

% Converts initial values into a single vector useable by ode45.
Y = zeros(4*P.Nodes, 1);
Y(1:P.Nodes) = initial(:,1);                    % x_m
Y(P.Nodes + 1 : 2*P.Nodes) = initial(:,2);      % y_m
Y(2*P.Nodes + 1 : 3*P.Nodes) = initial(:,3);    % x_c
Y(3*P.Nodes + 1 : 4*P.Nodes) = initial(:,4);    % y_c

% Sets simulation time.
Tmax = 5e4;
tspan = 0:Tmax;

% Solves ODE using ode45.
[T,Y] = ode45(@dYdt, tspan, Y);
fprintf('ODE solved.\n')

% Extracts membrane and cortex coordinates from vector output of ode45.
x_m = Y(:,1:P.Nodes);
y_m = Y(:,P.Nodes+1:2*P.Nodes);
x_c = Y(:,2*P.Nodes+1:3*P.Nodes);
y_c = Y(:,3*P.Nodes+1:4*P.Nodes);

% Extracts times at which links break.
links = LinkExtract(x_m,x_c,y_m,y_c,P.max_link,T,P.Nodes)';

% Plots membranes every 100 time steps. Commented out lines can be
% re-introduced to the code to save a movie of the evolution of the system.
if draw == 1
    lims = 1.05*[min(min(min(x_m),min(y_m))),max(max(max(x_m),max(y_m)))];
%     lims = [-15, 15];
    j = 100;
```

```
%     f = 1;
    for i = 1:length(T)
        if rem(i,j) == 0|| i == length(T)
            MembranesPlot(x_m(i,:),y_m(i,:),x_c(i,:),y_c(i,:),lims);
            text(0.9*lims(1),0.9*lims(2),['t = ',num2str(T(i)/1000),' s'])
            pause(0.01)
%             F(f) = getframe(gcf);
%             f = f+1;
        end
    end
% movie2avi(F,'movie.avi','compression','none','fps',floor(f/50));
end

% Plots only final configuration of membrane.
if draw == 2
    lims = 1.05*[min(min(min(x_m),min(y_m))),max(max(max(x_m),max(y_m)))];
    i = length(T);
    MembranesPlot(x_m(i,:),y_m(i,:),x_c(i,:),y_c(i,:),lims);
end

end
```

## B.2 Parameters_ode.m

```
function [P] = Parameters_ode ()
% Defines parameters such that they can be accessed by multiple functions
% and stores in a structure P.

P.Nodes = 80;          % Number of nodes in the system.
P.a = -5e-7;           % Tension coefficient.
P.b = 5e-11;           % Curvature coefficient.
P.k_m = 1e-3;          %\
P.k_c = 1e-2;          % } Elasticity constants for membrane, cortex and links.
P.k_l = 1e-2;          %/
P.max_link = 1.1;      % Maximum link length before link breaks.
P.P_Force = 0.42;      % Force providing isotropic pressure within cell.
P.min_sep = 0.35;      % Mimimum link separation before breaks.

end
```

## B.3 CircularInitial.m

```
function [initial] = CircularInitial (nodes)
% Creates matrix of specified size with circular 'membrane' contour that
% has number of nodes specified.

radius = ceil(nodes/(2*pi));         % Calculates appropriate radius.
initial = zeros(nodes,4);            % Initialises data matrix.
points = linspace(0,2*pi,nodes);     % Arranges nodes between 0 and 2pi.
sep = 0.9*radius;                    % Sets gap between cortex and membrane.
x_m = radius.*cos(points);  % \
y_m = radius.*sin(points);  %  } Converts points to Cartesian coordinates.
x_c = sep.*cos(points);     % |
y_c = sep.*sin(points);     %/
initial(:,1) = x_m;
initial(:,2) = y_m;
initial(:,3) = x_c;
initial(:,4) = y_c;

end
```

## B.4 InfInitial.m

```matlab
function [initial] = InfInitial (nodes)
% Creates matrix of specified size with squashed 'membrane' contour that
% has number of nodes specified.

radius = ceil(nodes/(2*pi));          % Calculates appropriate radius.
initial = zeros(nodes,4);             % Initialises data matrix.
points = linspace(0,2*pi,nodes);      % Arranges nodes between 0 and 2pi.
sep = 0.9*radius;                     % Sets gap between cortex and membrane.
undu = linspace(0,6*pi,nodes);        % Adds undulation to contour.
x1 = radius.*(cos(points) + 0.2.*cos(undu));     % \
y1 = radius.*(sin(points) + 0.2.*sin(undu));     %  } Converts points to
x2 = sep.*(cos(points) + 0.2.*cos(undu));        % |  Catesian coordinates.
y2 = sep.*(sin(points) + 0.2.*sin(undu));        %/
initial(:,1) = x1;
initial(:,2) = y1;
initial(:,3) = x2;
initial(:,4) = y2;
NumNodes = length(initial);                 % Simple check of number of nodes.

end
```

## B.5 dYdT.m

```matlab
function [Ydot] = dYdt(tspan,Y)
% Derives & defines odes for both the membrane and cortex.

% Define parameters for simulation.
P = Parameters_ode;
length_m = 1;
length_c = 1;
length_l = 0.5;
ds = 1/(P.Nodes);

% Extracts membrane and cortex coordinates from Y vector.
x_m = Y(1:P.Nodes);
y_m = Y(P.Nodes+1:2*P.Nodes);
x_c = Y(2*P.Nodes+1:3*P.Nodes);
y_c = Y(3*P.Nodes+1:4*P.Nodes);

% Creates vectors representing elements of initial to the left and right.
r = [2:P.Nodes,1];
l = [P.Nodes,1:P.Nodes-1];
rr = r(r);
ll = l(l);

% Calculates local normal to curve at each point.
[nx_m, ny_m] = LocalNormal(x_m', y_m', l, r);
nx_m = nx_m';    % } Ensures dimensions match.
ny_m = ny_m';    %/
[nx_c, ny_c] = LocalNormal(x_c', y_c', l, r);
nx_c = nx_c';    % } Ensures dimensions match.
ny_c = ny_c';    %/

% Calculates distances between nodes for membrane and cortex.
sx_m = (x_m(r) - x_m(l))/2;
sy_m = (y_m(r) - y_m(l))/2;
s_m = sqrt(sx_m.^2 + sy_m.^2);
sx_c = (x_c(r) - x_c(l))/2;
sy_c = (y_c(r) - y_c(l))/2;
s_c = sqrt(sx_c.^2 + sy_c.^2);

% Approximates 2nd and 4th order x and y derivatives by finite difference.
[d2x_m, d2y_m, d4x_m, d4y_m] = FinDif_24(ds,l,ll,r,rr,x_m,y_m);
[d2x_c, d2y_c, d4x_c, d4y_c] = FinDif_24(ds,l,ll,r,rr,x_c,y_c);

% Computes isotropic Pressure from surface area of membrane.
Area_m = sum(s_m);
Prs = P.P_Force/Area_m;

% Adds relaxation force of Hookean springs.
R_m = P.k_m * (s_m - length_m);
R_c = P.k_c * (s_c - length_c);

% Adds link interaction as Hookean spring.
x_l = x_m - x_c;
y_l = y_m - y_c;
s_l = sqrt(x_l.^2 + y_l.^2);
FL_x = P.k_l .* (s_l - length_l) .* (x_l ./ s_l); % (x_l./s_l) = cos(theta)
```

```matlab
FL_y = P.k_l .* (s_l - length_l) .* (y_l ./ s_l); % (y_l./s_l) = sin(theta)

% Adds additional force to prevent looping of membranes.
Stf_m = Stiffness(P.Nodes,x_m,y_m,r,l);%*0; % } Not needed if cortex fixed.
Stf_c = Stiffness(P.Nodes,x_c,y_c,r,l);%*0; %/

% Adds small deforming force to change cortex shape.
D_y = zeros(P.Nodes,1);
push = 21;      % Defines node acted on.
targ = 9.5;     % Defines how much to push.
D_y(push) = 0.1 * (y_c(push) - targ);

% % Adds possibility of linkage breaking if stretched too far (after vesicle
% % has reached stable shape).  Not needed if stochastic term activated.
% if tspan > 5000
%     FL_x(s_l>P.max_link) = 0;
%     FL_y(s_l>P.max_link) = 0;
% end

% % Define initial protrusion point for manual linker breakage.
% brk =  51;        % Defines which linker to break.
% if tspan > 5000
%     FL_x(brk) = 0;
%     FL_y(brk) = 0;
% end

% % Additional minimum node separation, representing finite width of links.
% if tspan > 5000
%     FL_x(s_m < P.min_sep) = 0;    % } Simplified method of breaking linkers
%     FL_y(s_m < P.min_sep) = 0;    %/  with no consideration of neighbours.
% %     [FL_x,FL_y] = CrowdBreak(P.Nodes,l,r,s_m,FL_x,FL_y,P.min_sep);
% end

% Adds stochastic possibility of all linkers breaking or reforming.
if tspan > 5000
    probs = rand(P.Nodes,1)>0.99;           % Generates random numbers.
    linkers = (FL_x + FL_y)~=0;             % Defines intact linkers.
    outcome = xor(probs,linkers);
    outcome = outcome.*(s_l<P.max_link);    % Ensures linkers > max break.
    FL_x = outcome.*P.k_l .* (s_l - length_l) .* (x_l ./ s_l);  % } Breaks/
    FL_y = outcome.*P.k_l .* (s_l - length_l) .* (y_l ./ s_l);  %/  fixes.
end

% Differential Equations.
dxdt_m = -((P.a*d2x_m + P.b*d4x_m)*0.2 + (R_m + Stf_m - Prs).*nx_m + FL_x);
dydt_m = -((P.a*d2y_m + P.b*d4y_m)*0.2 + (R_m + Stf_m - Prs).*ny_m + FL_y);
dxdt_c = -((P.a*d2x_c + P.b*d4x_c) + (R_c + Stf_c).*nx_c);%*0;
dydt_c = -((P.a*d2y_c + P.b*d4y_c) + (R_c + Stf_c + D_y).*ny_c);%*0;
            % (Uncomment the *0 above to fix cortex in place).

Ydot = [dxdt_m; dydt_m; dxdt_c; dydt_c];    % Reforms vector for ode45.

end
```

## B.6 LocalNormal.m

```
function [nx, ny] = LocalNormal (x,y,l,r)
% Calculates local normal at all points on a curve, and returns vectors of
% the x and y components individually.

tangs = [x(r) - x(l); y(r) - y(l)];
norm = sqrt(tangs(1,:).^2 + tangs(2,:).^2);
nx = tangs(2,:) ./ norm;     % = cos(theta)
ny = -tangs(1,:) ./ norm;    % = sin(theta)

end
```

## B.7 FinDif_24.m

```
function [d2x,d2y,d4x,d4y] = FinDif_24 (ds,l,ll,r,rr,x,y)
% Utilises finite difference method to approximate second and fourth order
% derivatives of x and y vectors of contour.

% Approximates 2nd order x derivative.
d2x_l = (x(l) - x) / ds^2;
d2x_r = (x(r) - x) / ds^2;
d2x = d2x_l + d2x_r;

% Approximates 4th order x derivative.
d4x_ll = (x(ll) - x(l)) / ds^4;
d4x_l = d2x_l / ds^2;
d4x_r = d2x_r / ds^2;
d4x_rr = (x(rr) - x(r)) / ds^4;
d4x = d4x_ll - 3*(d4x_l + d4x_r) + d4x_rr;

% Approximates 2nd order y derivative.
d2y_l = (y(l) - y) / ds^2;
d2y_r = (y(r) - y) / ds^2;
d2y = d2y_l + d2y_r;

% Approximates 4th order y derivative.
d4y_ll = (y(ll) - y(l)) / ds^4;
d4y_l = d2y_l / ds^2;
d4y_r = d2y_r / ds^2;
d4y_rr = (y(rr) - y(r)) / ds^4;
d4y = d4y_ll - 3*(d4y_l + d4y_r) + d4y_rr;

end
```

## B.8 Stiffness.m

```matlab
function [Stf] = Stiffness(Nodes,x,y,r,l)
% Creates additional force term to ensure the nodes never bend by less than
% pi/2.  Uses A.B = |A||B|cos(theta).

Stf = zeros(Nodes,1);
AB = (x(l) - x).*(x - x(r)) + (y(l) - y).*(y - y(r));    % A.B
A = sqrt((x(l) - x).^2 + (y(l) - y).^2);                 % |A|
B = sqrt((x - x(r)).^2 + (y - y(r)).^2);                 % |B|
theta = acos(AB./(A.*B));    % Calculates external angle between vectors.
phi = pi - theta;            % Calculates internal angle.
Cos_phi = 0.01*cos(phi);
Stf(phi<(pi/2)) = Cos_phi(phi<(pi/2));  % Assigns force for extreme angles.

end
```

## B.9 CrowdBreak.m

```matlab
function [FL_x,FL_y] = CrowdBreak(Nodes,l,r,s_m,FL_x,FL_y,min_sep)
% Breaks links in areas of high link density on the membrane such that
% there is a minimum separation representing the physical width of the
% links.  To allow for links relaxing into space vacted by other links that
% have broken, it is randomly decided whether odd or even nodes are
% processed first, with logical operations concluding the calculation.

% Creates logical vectors representing odd and even elements.
elements = (1:Nodes)';
odd = zeros(Nodes,1);
odd(rem(elements,2) ~=0 ) = 1;
even = zeros(Nodes,1);
even(rem(elements,2) == 0) = 1;

% Assigns odd and even operations to a random order.
rnd = rand(1);
if rnd > 0.5
    first = odd;
    second = even;
else
    first = even;
    second = odd;
end

% Creates logical map of which links are to be broken.
not_broken = first;
not_broken(s_m < min_sep) = 0;
not_broken = not_broken + second;
spaces = not(not_broken(l) & not_broken(r));
not_broken(s_m < min_sep) = 0 + spaces(s_m < min_sep);

% Breaks links.
FL_x = not_broken.*FL_x;
FL_y = not_broken.*FL_y;

end
```

## B.10 LinkExtract.m

```
function [links] = LinkExtract(x_m,x_c,y_m,y_c,max_link,T,Nodes)
% Finds the time at which each extra link is broken.

% Finds time at which each link that breaks is broken.
links = zeros(Nodes,3);
for t = 5000:length(T)  % Ensure t start point matches time link cut.
    s_l = sqrt((x_m(t,:)-x_c(t,:)).^2 + (y_m(t,:)-y_c(t,:)).^2);
    broken = sum(s_l > max_link);
    if broken > 0
        if links(broken,2) == 0
            links(broken,:) = [broken, t, T(t)];
        end
    end
end

% Removes empty elements where more than one link breaks at the same time,
% or links that don't break.
links2(1,:) = links(links(:,1)~=0,1);
links2(2,:) = links(links(:,2)~=0,2);
links2(3,:) = links(links(:,3)~=0,3);
links = links2;

end
```

## B.11 MembranesPlot.m

```
function [] = MembranesPlot(x_m,y_m,x_c,y_c,lims)
% Plots membrane and cortex, as well as links if specified.

figure(1)
clf
hold on
plot(x_m,y_m, '-b')      % Plots membrane.
plot(x_c,y_c, '-r')      % Plots cortex.
plot(x_m([80,1]),y_m([80,1]),'-b')  % } Closes gap in plot.
plot(x_c([80,1]),y_c([80,1]),'-r')  %/

% Extracts values from parameters file.
P = Parameters_ode;

% Calculates linker lengths and separations for plotting check.
s_l = sqrt((x_m-x_c).^2 + (y_m-y_c).^2);
% r = [2:P.Nodes,1];             % } Activate (plus line below) to not plot
% l = [P.Nodes,1:P.Nodes-1];     %/  linkers closer than minimum separation.
% s_m = 1/2 * sqrt((x_m(r) - x_m(l)).^2 + (y_m(r) - y_m(l)).^2);s

% Plots all linkers aren't broken.
for i = 1:P.Nodes
    if (s_l(i) < P.max_link) %&& (s_m(i) > P.min_sep)
        plot([x_m(:,i),x_c(:,i)],[y_m(:,i),y_c(:,i)], '-g')
    else
        text(x_m(:,i),y_m(:,i),num2str(i))  % Identifies broken linkers.
    end
end

xlim ([lims(1) lims(2)])
ylim ([lims(1) lims(2)])
axis square
% legend('M','C','L')
hold off

end
```