ENGINEERING PHYSICS

Bachelor's thesis

A computational model for the cooperative binding and self-organization of Myosin-VI in membrane reshaping.

Pau Blanco Dorca

Supervisor:

Prof. Dr. Erwin Frey
(LMU Munich)

Co-director:

Dr. Sergio Alonso Muñoz
(UPC)

Advisor:

Laeschkir Würthner
(LMU Munich)

June 2020

Acknowledgements

I would like to thank Laeschkir Würthner for this guidance throughout this work, also Prof. Dr. Erwin Frey for accepting me to the Statistical and Biological Physics Group in Munich. And I would also like to thank all the members of the group for the welcoming attitude during my brief stay.

Abstract

The dynamic membrane reshaping processes are vital for the cells to preform its activity normally. They are involved in a myriad of cellular pathways and can be preformed by different agents. One of those is the myosin-VI, one of the motor proteins that produce the force that will move the membrane. To do that the protein binds to the membrane, this binding has been seen to be targeted towards specific regions of the membrane.

In this project a model for the cooperative binding of the myosin VI and a membrane pore will be developed. The growth of the pore will produce a partial derivative equation with moving boundaries that will be solved numerically utilizing the Boundary element method. Then the model will be validated by comparing it to experimental data.

Contents

1 Introduction **2**
1.1 Objectives and work done 3

2 Cell biology background **5**
2.1 Cell membrane 5
2.2 Membrane curvature reshaping 6
2.3 Motor proteins 6
2.3.1 Myosin-VI 8

3 Experimental setup **11**

4 The model **14**
4.1 Linear stability analysis 17

5 Mathematical derivation of the numerical solution **23**
5.1 The Laplace Equation 23
5.2 Reciprocal relation 24
5.3 Kernel function 25
5.4 Boundary integral solution 26
5.5 Boundary solution with constant elements 27
5.6 Evolution of the curve 30

6 Computer implementation **32**

7 Results and discussion **35**
7.1 Concentration dependency 35
7.2 Pore growth 36

8 Conclusions and further work **39**

9 Appendix: Code **41**
9.1 Matlab 41
9.2 C++ 47

1 Introduction

Molecular and cell biology are the perfect targets for the application of computational models. In these fields, the infinity of processes and interactions between the different biomolecules of the cell are of a complexity that makes them impossible to model with a exhaustive representation of all that is happening. Every broken hydrogen bond or the way every protein is folded to preform a certain action would require an astronomical amount of time to compute, so seldom the solution is to use mathematical models that unite millions of different molecular processes to a single variable. This allows for the modelization of complex processes without losing the important features about them.

Another degree of complexity is added when we take into account the irregular geometry where all this cellular processes take place. Even the more simple physical phenomena can present emergent behaviours when they occur in a geometrically complex space. This makes mathematical models, that could be easily solved by hand in a regular or highly symmetric environment impossible to solve analytically. In order to obtain results we must turn to the numerical methods and the computer simulations. These methods will be able to solve any model that can be discretized for a computer to do the calculations for us.

For this work we wanted a biological problem that would benefit from a computational approach in order to solve it. This is the reason why the growth of a membrane pore triggered by cooperative the binding of the Myosine-VI was chosen. The pores that emerge from this interaction between the membrane lipids and the motor protein do not have a regular and symmetric shape, they instead present irregular shapes that are impossible to calculate analytically.

With the huge advances in the computational numeric simulations along the years a great amount of software has been presented to solve all of the typical mathematical problems that can be encountered. This ranges from the simulation of all kinds of fluids to the computation of the structural integrity of solids of a multitude of shapes, among many others. Although these programs are a really useful toolkit, one of the objectives of this thesis was to create the code for the partial differential equation solver by ourselves in order to gain a greater insight in the way it works,

both analytically and numerically. Normally for a simple two dimensional partial differential equation can be solved using any commercial software without much more thought. And this is why the model of a growing pore in a cellular membrane was chosen. This problem has the peculiarity that the shape of it's boundary evolves with time. This means that the simulation had to be build from the start to be able to freely move the boundary and a way to move this boundary needed to be included.

1.1 Objectives and work done

The first objective was to develop the model, based on the results from the experimental work of [6]. This would let us see which are be the mathematical tools that will be needed. To do that, different models for diffusion limited aggregation and the solidification where studied[8] and [7], due to its geometrical similarity with the myosin binding process. Even though they belong to very different areas the growth of the interfaces observed in the laboratory resembled the processes depicted in the studied papers. With this base a model was devised.

The parameters of the obtained model needed to be fitted to the experimental results and in order to do that we started deriving the linear stability analysis. The results from this analysis helped us choose the values that would lead to the instabilities that appeared in the experiments.

Then we discretized the equations from the model using the boundary element method. This would allow us to solve the partial differential equations present in the model. Other numerical methods to solve the other parts of the problem, such as integrals or derivations of different variables that appear in the model. Such as finite differences or Euler integration. Even though these methods are amongst the simpler ones for the integration and derivation of sampled functions they are fast and since they are to be preformed several times every time step the velocity of the method was an important factor.

Once the mathematical derivations where done the model needed to be implemented in a computer in order to solve it. Since this simulation would require an important amount of operations the core language had to be a compiled one

because of its speed compared to interpreted ones such as Python or Matlab. The chosen language was C++ because of its versatility and the libraries that could be used to solve some parts of the model, such as the linear systems created by the boundary element method and the option to use parallelism to make the project faster.

Once the code was working we extracted the results. Because of the numeric demand of this simulation we obtained the variations of the pore growth with he concentration with a simple circular pore. And then we preformed various simulations to determine the different parameters that generated the desired pores.

2 Cell biology background

2.1 Cell membrane

Lipids or fatty acids are composed of a hydrophilic head and an hydrophobic tail. This asymmetry leads to the auto-organization of groups of lipids when they are in an aqueous medium. The cell membrane is a lipidical bilayer, in other words, a double sheet of fatty acids arranged in order for the hydrophobic tail end to be in between the layers and the hydrophilic head pointing outside. The different lipids are united between them by non-covalent bonds, much weaker than the bonds that form each biomolecule. These weaker bonds allow the components of the membrane to laterally diffuse along the membrane leaflet, which makes membranes a two-dimensional fluid. This diffusivity depends on the composition of the lipids of the membrane and is different for different parts of the membrane or for the membranes of the different organelles. Membranes also contain a myriad of proteins that preform different functions. These proteins control the rate at which certain molecules go through the bilayer and provide binding sites for other proteins to attach to the membrane, among other functions.[**cell**]

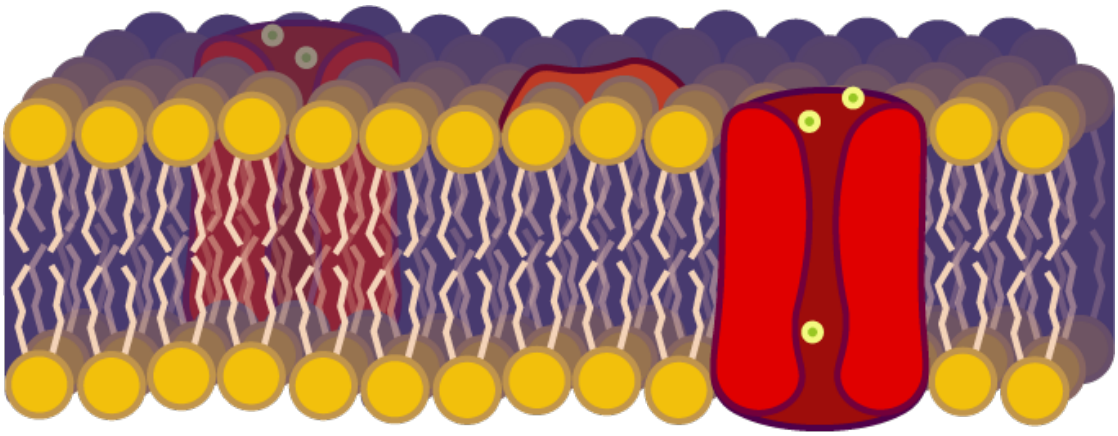

Figure 1: Drawing of the membrane, formed by two layers of lipids and the membrane proteins, in red

This structure is not exclusive to the cell boundary, it can also be found in different organelles of the cell such as the Golgi apparatus or the mitochondria among others. These organelles have different shapes to fulfill each purpose, these diverse

morphologies need different membrane curvatures, and these curvatures have to be dynamically changed in order to carry on the different cellular processes in each organelle.

2.2 Membrane curvature reshaping

For pathways such as cell motility and vesicle trafficking the membrane of the cell and some of its organelles must be reshaped dynamically as the process advances. This is a complex process that is carried on by the joint work of different proteins, both embedded in the membrane and exterior ones, lipids and other agents such as actin filaments and other elements of the cytoskeleton. These biomolecules together create a machinery that can sense and reshape the curvature of the membrane to preform the required tasks.

This dynamic reshaping can be generated by different agents that work together to obtain different degrees and orientations of membrane curvature. For example by changing the composition of certain membrane lipids bigger heads can be obtained or the expansion of the acyl tails can induce a curvature,[**curvature**]. Transmembrane proteins such as potassium channels can also act as a wedge that favours certain curvatures because of its asymmetry. This makes the leaflet with the thinner part of the protein a bit smaller and induces a curvature.

The cytoskeleton and motor proteins are also primordial for this process. The versatility of the actin cytoskeleton is key to the creation of high curvature regions. This process, like some of the previous ones, requires mechanical force to take place. This force is provided by the motor proteins that move along the citoskeletal fibers and can bind to the membrane.

2.3 Motor proteins

Motor proteins differ from other proteins because they can convert the energy obtained from the ATP hydrolysis into movement along a polymer, such as actin or tubulin. There are three different superfamilies of motor proteins, the kinesins and dyneins that move along tubulin filaments, and myosin molecules that additionally move along actin[3].

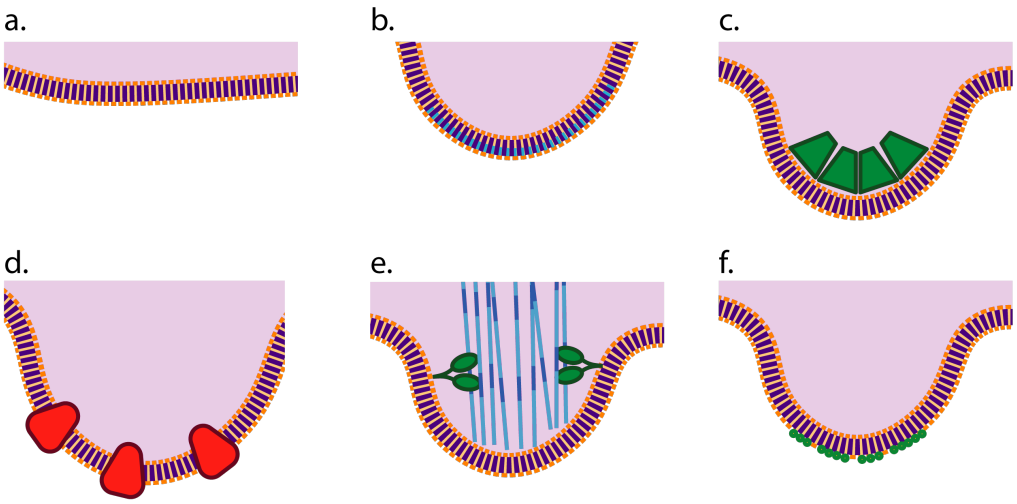

Figure 2: Different processes by which membrane curvature can be achieved. First we have the original piece of membrane without curvature. One of the ways to induce a curvature to the membrane would be by changing the composition of the tails of the lipids, making them occupy more space in one leaflet than in the other. The second method would be by external scaffolding, where coating proteins are pushed against the membrane to induce and maintain the desired curvature. Membrane proteins can also produce a curvature by its shape that will make the outer leaflet bulge out. This widening of the outer leaflet can also be accomplished by changing the composition of the heads of the lipids. Finally this curvature can be obtained by pushing the actin filaments towards the membrane.

The first two groups have a higher procession. This means that these proteins will move a greater distance along the filament before detaching. This makes them useful for tasks that require a movements from the center of the cell to the membrane or the other way around. For example the transport of vesicles or other small organelles and macro molecules for relatively long distances, at a sub-cellular scale.

The myosin molecules, on the other hand, move smaller distances along the filament before detaching. This makes them low procession motors[4] and makes

them useful for other functions, such as the contraction of the muscle or duties along the cortical actin network underneath the plasma membrane. These normally include some type of dynamic membrane reshaping, which is the focus of this work. This can be done by different methods, by attaching directly to the membrane by a protein-lipid interaction and then move it with respect to an actin filament to create a shape that will be useful for a particular process.

2.3.1 Myosin-VI

Inside of the family of myosin motor proteins there are different subtypes, theses have the same head sector but differ in in the number and disposition of the tails[2]. This makes them specialized for different applications. This protein can be found in the great majority eukaryotic cells. Although the rest of myosin molecules move towards the the positive end of the actin filaments, this particular subtype moves in the other direction. Since the actin filaments are theorized to have their plus ends generally towards the plasmatic membrane this differential feature makes it crucial for various processes in multiple places of the cell membrane where the myosin-VI will move away from it unlike the rest of myosin proteins. It is also key in some interactions in the Golgi complex, where the filaments point towards organelle, then only a motor that goes in the minus direction could move away from it.

In particular it is present in vesicular membrane traffic, cell migration and autophagy among others. In vesicular membrane traffic the myosin VI would help with the creation of the vesicles by pulling the membrane of the cell or the organelle. This creates a bud that will end in a vesicle that will be pushed away from the membrane also carried by bound myosin molecules moving along an actin track away from the organelle or membrane.

In cell migration the myosin attaches to the membrane and by staying in the same place moves the actin filaments, providing to them the force to push the membrane towards the direction of the movement forming a protrusion, or deformation in the cell. This protrusion will create a tension in the whole cell that will pull it towards the direction of the actin push. Since myosin-VI is the only actin motor that moves

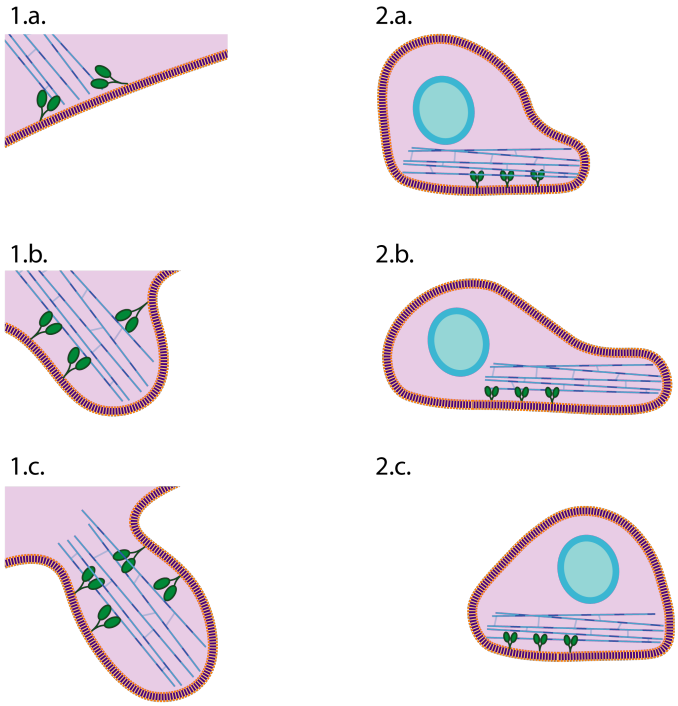

Figure 3: Different processes where the myosin-VI plays a key role. On the first column we can observe the budding process. Where a vesicle is creaed by pushing the actin filments towards the membrane. In the second column we can see cell migration, where myosin generates the protrusion that pulls the cell towards a certain direction.

away from the membrane it is also the only one that will be able to push the actin filaments towards the membrane and create this movement.

In autophagy, where the cell encapsulates elements of the cytoplasm into a vesicle named autophagosome to metabolize them. This can be done either to neutralize dysfunctional or infectious elements or to obtain nutrients when there is no other source, myosin VI helps with the creation of this vesicle. Then this vesicle is fused with a lysosome, which contains the enzymes that will digest its contents, in this process myosin also helps by bringing the lysosome close to the autophagosome.

In all this different processes the complex interactions between the protein head of the myosin and the different binding sites and lipids of the membrane are what

determines where and at which rate the myosin attaches to the membrane. Then by studying the behaviour of the myosin in different conditions it would be possible to characterize this interactions. This would allow to better understand and maybe control this processes. In order to do that an experimental setup has been devised in which we can control some of the parameters and see how they affect the binding of the myosin to the lipid bilayer.

3 Experimental setup

In order to study the motor-lipid interaction a experimental setup was studied in [6]. It consisted of a synthetic membrane, which was a phosphatidylcholine bilayer, present on a great number of eukaryotic cells. This membrane was grown over a flat mica substrate. It is in contact with a solution that emulates the cytosolic conditions with myosin-VI and actin filaments among other components. The membrane stiffness could be changed in future setups to observe which are the changes in the interactions between the membrane and the motor proteins. The results of these experiments showed that the binding of the myosin to the membrane was sparse but highly cooperative. A small defect in the membrane

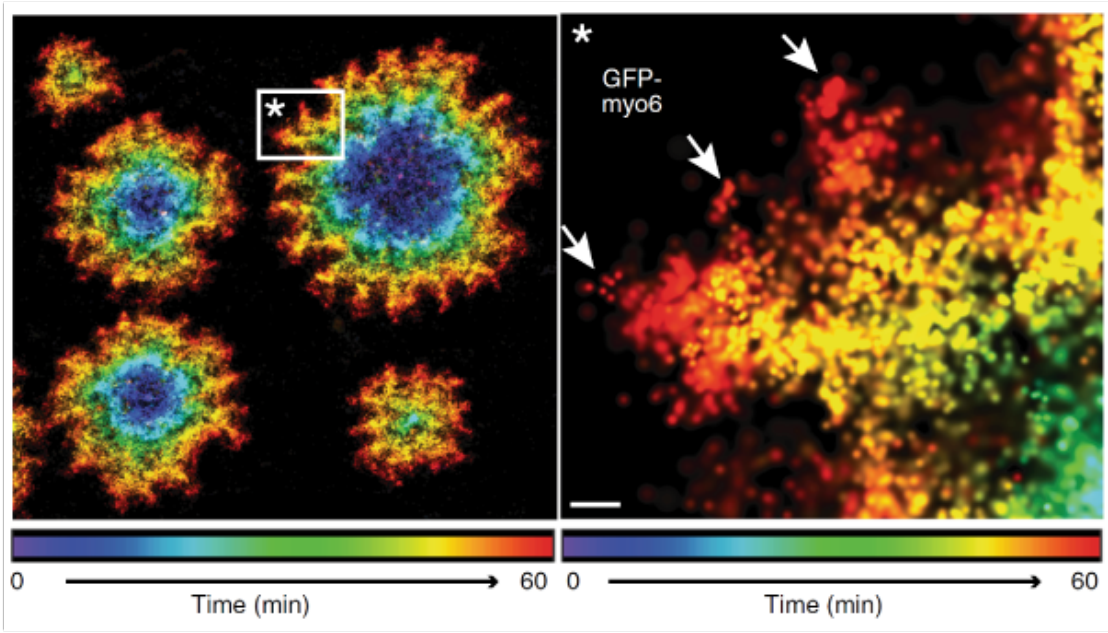

Figure 4: Example of pore growth for a concentration of myosine VI of 150nM. The pores where grown for 60 minutes We can observe the apparition and splitting of the hotspots, with a higher concentration of myosin due to the cooperative binding enhanced by the curvature

would led to the binding of the first proteins and these would make the defect grow, creating a curvature in the membrane that would allow for more proteins to attach to the growing pore. This cooperativity leads to the formation of flower-like patterns with a regular distance between the hotspots of high concentration of

myosin-VI. This patterns keep growing as long as there is material in the solution to bind to the membrane. It is also seen in the experiments that the surplus of lipids that are pushed out of the membrane create small vesicles near these pores. The results from these setups can be seen in Fig. 5 and they will be used as a

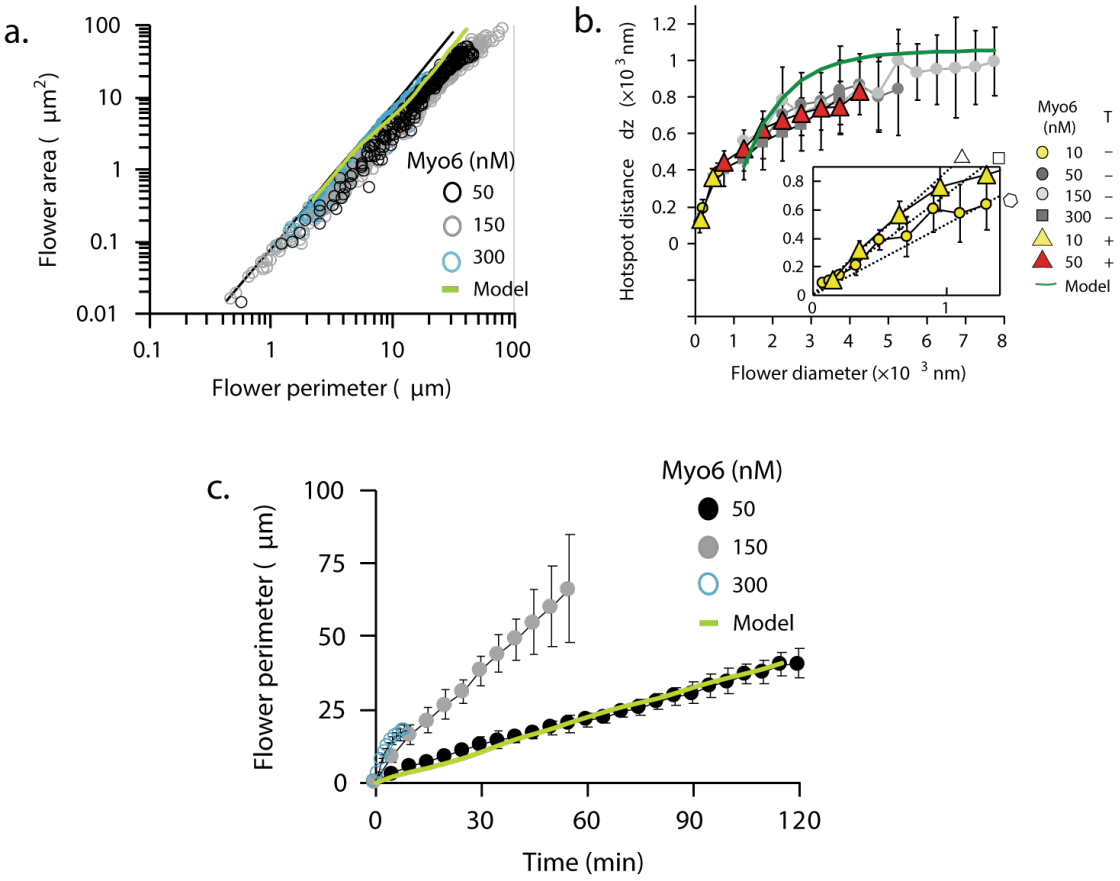

Figure 5: Results of the experiments that define the key aspects that the model needs to reproduce. The first sub-figure illustrates that the relation between the flower perimeter and the flower area is a power law, and it is the same for the different studied concentrations. In the second plot we can see that the hotspot distance is bounded to $1\mu m$ with diameters of up to $8\mu m$. For this feature to be present in the model higher frequencies need to appear as the radius grows. Finally in the last figure we can see the effect of the concentration of myosin in the bulk in the velocity of growth. As the concentration goes from $50nM$ to $150nM$ the growth of the perimeter is also triplicated. All these aspects need to be included in the model.

starting point for developing the model for this work. These results will also give

us a template to compare the results of the model and fit the parameters that cannot be directly extracted from the experiments. We can look for the distance between hotspots or the evolution of the perimeter and the area of the interface to see if the model accurately represents this process. Once we are certain of that we can change the parameters of the model, such as the bulk diffusion or the concentration of myosin molecules to see how these changes affect the growth of these membrane pores.

4 The model

After studying the results from the experiment we start to design the model, the important features that this needs to include are the cooperative effect that the curvature of the membrane has on the binding of the myosin and the hotspot distance, that is maintained over the growth once a certain radius is reached. In order to maintain this distance in a growing perimeter we will need peaksplitting instabilities to arise as the radius increases. This model is based on diffusion limited solidification models like [8] or[7].

In order to represent the attachment of the myosin-VI to the interface and the evolution of the interface itself observed in the experimental setup we start by defining the interface of the pore as a closed one dimensional curve $\mathbf{C}$,in R^2. Then we will define the concentration of myosin in the curve. As seen in the experiments and posed in the paper[6], this concentration will depend on the curvature of the interface. A greater outwards curvature will imply that more myosin will be able to bind to the interface, as shown in Fig. 4. This dependency will be given by the line tension coefficient τ that will establish how strongly the curvature, $\kappa(\boldsymbol{q})$ in the equations into the concentration at the interface.

$$c(\boldsymbol{q}) = c_0(1 + \tau\kappa(\boldsymbol{q})) \qquad \forall \boldsymbol{q} \in \mathbf{C} \tag{1}$$

The value c_0 will be a fraction of the bulk concentration c_∞ and its value will be determined with the linear stability analysis.

In the bulk the myosin will be moving freely in a solution. To simulate this we will assume that the concentration of myosin will diffuse with a coefficient D. Then, since we are only interested in the part that is in contact with the membrane we will project this three dimensional bulk into a two dimensional surface over the lipid bilayer. This diffusion coefficient will be given by the solution used to emulate the cytosol in the experiment. This diffusion has to be defined in a domain, $\mathbf{R} \in R^2$ that contents the exterior of $\mathbf{C}$, and will be reigned by the expression

$$\partial_t c(\boldsymbol{q}) = D\nabla^2 c(\boldsymbol{q}) \qquad \forall \boldsymbol{q} \in \mathbf{R} \tag{2}$$

Then we need to include the effect of force that the myosin applies to the mem-

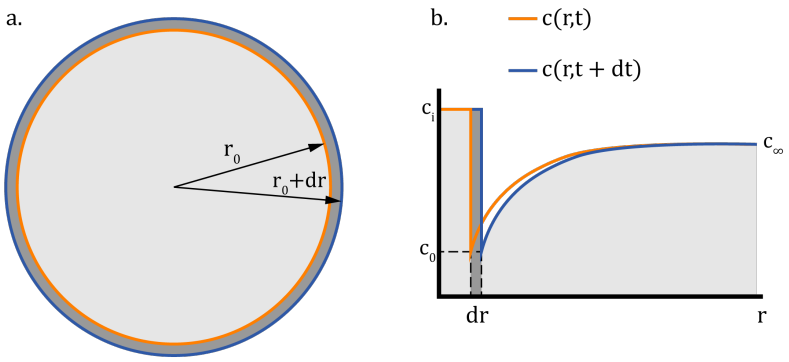

Figure 6: Radial concentration profile(b.) created by a circular pore (a.) with the depletion zone near the interface at r_0 and the evolution of the concentration profile for a time differential

brane together with the redistribution of the proteins along the pore interface. In order to obtain this equation we will look at a simple pore, a circular one. We will have the concentration profile shown in Fig. 6, with a depletion zone near the membrane pore due to the binding of the cytosol myosin to the pore interface, leading to the minimum concentration of c_0 right before the interface. The parameter c_i takes into account the proteins integrated itno the interface. The number of particles must be conserved, so the number of particles attaching to the interface will determine automatically its growth of the area covered by the interface. Given by the next equation

$$D\partial_r c|_{r_0} = (c_i - c_0)\partial_t A \tag{3}$$

Then we use that for a circumference the area differential is defined as $dA = 2\pi r dr$. Then,we define the normal velocity of the surface as the derivative of respect of the radius. Then, for a circumference of radius r_0, Fig. 6 it will be $v_n = \partial_t r_0$. Finally we assume that the concentration in the membrane interface is much larger than the one in the bulk near it, $c_i \gg c_0$. This is reasonable because of the high cooperativity that the membrane presents. We then obtain the equation for the normal velocity of the interface.

$$v_n = \frac{D}{c_i}\partial_n c|_{r_0} \tag{4}$$

With this final condition we would have a sufficient model to describe the results of the experiment. Which put together reads

$$c(\boldsymbol{q}) = c_0(1 + \tau\kappa(\boldsymbol{q})) \qquad \forall \boldsymbol{q} \in \mathbf{C} \tag{5}$$

$$\partial_t c(\boldsymbol{q}) = D\nabla^2 c(\boldsymbol{q}) \qquad \forall \boldsymbol{q} \in \mathbf{R} \tag{6}$$

$$c(q) = c_\infty \qquad |\boldsymbol{q}| \to \infty \tag{7}$$

$$v_n(\boldsymbol{q}) = \frac{D}{c_i}\partial_n c(\boldsymbol{q}) \qquad \forall \boldsymbol{q} \in \mathbf{C} \tag{8}$$

This model can be further simplified by observing the parameters of the problem. We compute the Péclet number for this equations, this dimensionless number relates the advection and diffusion terms. To do this we compare the timescale of diffusion

$$t_{diff} \approx \frac{r_0^2}{D} \tag{9}$$

The advective term of this problem will be the velocity at which the interface moves, and its timescale will be

$$t_{growth} \approx \frac{r_0}{v_n} \tag{10}$$

Then the dimensionless Péclet number will be the rate between the two timescales

$$Pe_{r_0} = \frac{v_n r_0}{D} \tag{11}$$

If we input typical values to the parameters obtained from the experiments[6] and bibliography. Such as $r_0 = 1\mu m$, $v_n \sim 10^{-3}\frac{\mu m}{s}$ and $D = 1\frac{\mu m^2}{s}$we obtain a value of $Pe \sim 10^{-3}$. This means that the diffusion several orders of magnitude faster than the pore growth. Then we will assume that at every time step of the simulation the concentration on the cytosol has already reached the equilibrium position before we evolve the interface. From this result we can make the approximation from [2] to a Laplace equation for the three dimensional bulk projected into the two dimensional plane where the membrane is.

$$0 = \nabla^2 c(\boldsymbol{q}) \qquad \forall \boldsymbol{q} \in R \tag{12}$$

With this simplification we will now preform a linear stability analysis to the system in order to see which combinations of parameters lead to a band of unstable modes that will start the growth of the instabilities in the pore interface.

4.1 Linear stability analysis

In order to study the stability of our system we will slightly perturb a circular interface with different modes to see its linear evolution[5]. First we search for the different modes that will fit this perturbation.

We start with the 2D Laplace equation in polar coordinates

$$\frac{1}{r}\partial_r \left[r\partial_r f(r,\theta)\right] + \frac{1}{r^2}\partial_\theta^2 f = 0 \tag{13}$$

Then we assume that the solution can be obtained by separation of variables. So the function we are looking for will be of the form $f(r,\theta) = R(r)\Theta(\theta)$. If we put this function into the Laplace equation in polar coordinates we obtain

$$\Theta\partial_r^2 R + \Theta\frac{\partial_r R}{r} + \partial_\theta^2\Theta\frac{R}{r^2} = 0 \tag{14}$$

By reordering the equation we obtain

$$r^2\frac{\partial_r^2 R}{R} + r\frac{\partial_r R}{R} = -\frac{\partial_\theta^2\Theta}{\Theta} \tag{15}$$

Since this equality holds for any point we can assume that both parts are constant, so we define the constant n, that will determine the mode we are studying

$$r^2\frac{\partial_r^2 R}{R} + r\frac{\partial_r R}{R} = n^2 \qquad \frac{\partial_\theta^2\Theta}{\Theta} = -n^2 \tag{16}$$

Both differential equations can be easily solved and we obtain

$$R(r) = r^{-n} \qquad \Theta(\theta) = \cos n\theta \tag{17}$$

Which are not the unique solutions but are enough for the problem at hand, but they are enough to describe the modes that we will use. If we join both the radial

and the angular parts we obtain a series of solutions that will be named P_n

$$P_n(r,\theta) = r^{-n}\cos n\theta \tag{18}$$

Then we can define the perturbed interface as a circle with a modulation defined by the mode

$$r(\theta) = r_0 + \delta\cos n\theta \tag{19}$$

With r_0 the radius of the initial circular interface and δ the amplitude of the small perturbation, it is assumed that $r_0 \gg \delta$.

Once we have defined the interface we can calculate the boundary condition as stated in the model Eq. (1). For this we need to calculate the curvature of the interface, which we can obtain using the following formula, where the dot notation refers to the derivative with respect to θ

$$\kappa(\theta) = \frac{|r^2 + 2\dot{r}^2 - r\ddot{r}|}{(r^2+\dot{r}^2)^{3/2}} \tag{20}$$

Since we are working with a slightly perturbed sphere we will consider the terms of the derivative squared to be negligible against the squared value of the radius and that the second derivative of the radius will be always smaller than the radius itself. With this approximations we obtain

$$\kappa(\theta) \approx \frac{r^2 - r\ddot{r}}{r^3} = \frac{1}{r} - \frac{\ddot{r}}{r^2} \tag{21}$$

Then we can approximate the first term around r_0 to the first order of δ

$$\frac{1}{r} = \frac{1}{r_0} - \frac{1}{r_0^2}(r - r_0) = \frac{1}{r_0}\left(1 - \frac{\delta\cos n\theta}{r_0}\right) + O(\delta^2) \tag{22}$$

Then we can compute $\ddot{r}$ by using Eq. (19)

$$\ddot{r} = \delta\cos n\theta = -n^2\delta\cos n\theta \tag{23}$$

With this result we can also approximate the second term

$$\frac{-n^2\delta\cos n\theta}{r^2} \approx -n^2\delta\cos n\theta\left(\frac{1}{r_0^2}-\frac{2}{r_0^3}(r-r_0)\right) = \frac{-n^2\delta\cos n\theta}{r_0^2}\left(1-\frac{2\delta\cos n\theta}{r_0}\right) \approx -\frac{n^2\delta\cos n\theta}{r_0^2} \tag{24}$$

Once we have done all the approximations to the first order of δ we obtain the curvature

$$\kappa(\theta) \approx \frac{1}{r_0}\left(1-\frac{\delta\cos n\theta}{r_0}\right)+\frac{n^2\delta\cos n\theta}{r_0^2} = \frac{1}{r_0}\left(1+\frac{\delta\cos n\theta}{r_0}\left(n^2-1\right)\right) \tag{25}$$

Then by using this result and Eq. (1) the concentration in the interface **C** will be

$$c_{\mathbf{C}}(\theta) \approx c_0\left[1+\frac{\tau}{r_0}\left(1+\frac{\delta\cos n\theta}{r_0}(n^2-1)\right)\right] \tag{26}$$

Then we can compute the solution on the rest of the domain **R**. In order to do this we have to provide a solution of the Laplace equation that will be equal to Eq. (26) at the interface and also fulfill the condition $c(r_1) = c_\infty$ with $r_1 \gg r_0$.

This solution will have the general form of

$$c(r,\theta) = A\ln r + B\delta P_n(r,\theta) + C \tag{27}$$

By expanding it to the first order of δ around the interface we obtain

$$c_C(\theta) = A\left(\ln r_0+\frac{\delta\cos n\theta}{r_0}\right)+\frac{B\delta\cos n\theta}{r_0^n}+C \tag{28}$$

If we equal this last equation to Eq. (26) and use the boundary condition for the bulk we obtain the three coefficients

$$A = \frac{1}{l}\left[G+\frac{c_0\tau}{r_0}\right] \qquad B = r_0^{n-1}\left[\frac{mc_0\tau}{r_0}\left[n^2-1-\frac{1}{l}\right]-\frac{G}{l}\right] \qquad C = c_\infty - A\ln r_1 \tag{29}$$

With

$$G = c_0 - c_\infty \qquad l = \ln\frac{r_0}{r_1} \tag{30}$$

Once we have the concentration field we have to compute its normal derivative in

the interface to see the evolution of its perimeter. Since the interface has an almost circular shape we will approximate the normal derivative as a radial derivative $\partial_r c$. Then by taking the the radial derivative of Eq. (27) we obtain

$$\partial_r c = \frac{A}{r} - \frac{Bn\delta\cos n\theta}{r^{n+1}} \tag{31}$$

And by approximating it around r_0 we obtain

$$\partial_r c \approx \frac{1}{lr_0}\left[G + \frac{c_0\tau}{r_0}\right] - \frac{\delta\cos n\theta}{r_0^2}\left[\frac{nc_0\tau}{r_0}\left[n^2-1\right] + \left[G+\frac{c_0\tau}{r_0}\right]\frac{1-n}{l}\right] \tag{32}$$

Then by using the definition of the normal velocity of the interface given in the model we obtain

$$v_n = \frac{D}{c_i}\left[\frac{1}{lr_0}\left[G + \frac{c_0\tau}{r_0}\right] - \frac{\delta\cos n\theta}{r_0^2}\left[\frac{nc_0\tau}{r_0}\left[n^2-1\right] + \left[G+\frac{c_0\tau}{r_0}\right]\frac{1-n}{l}\right]\right] \tag{33}$$

This result gives us the velocity of the whole interface, but we can extract the velocity of a mode by taking the time derivative of Eq. (19)

$$v_n = \frac{dr(\theta)}{dt} = \frac{dr_0}{dt} + \frac{d\delta}{dt}\cos n\theta \tag{34}$$

This result lets us separate the growth of the unperturbed circle and the growth of the perturbation. The radius of the circle will grow following the next expression

$$\dot{r}_0 = \frac{D}{lc_i r_0}\left[G + \frac{c_0\tau}{r_0}\right] \tag{35}$$

This result yields that the initial interface needs to have a minimum radius in order to start its growth, this critical radius r_c is given by

$$r_c = \left[\frac{c_0\tau}{c_1 - c_0}\right] \tag{36}$$

And the evolution of the perturbation amplitude will be given by the next expression

$$\frac{\dot{\delta}}{\delta} = -\frac{D}{c_i r_0^2}\left[\frac{nc_0\tau}{r_0}\left[n^2-1\right] + \left[G+\frac{c_0\tau}{r_0}\right]\frac{1-n}{l}\right] \tag{37}$$

Once we know the linear behaviour we can obtain the dispersion relation for different groups of parameters, this will help us chose the correct parameters for the simulations. We start by setting some of the known parameters from the experiments. We will set the concentration from a range around 50nM to 150nM. Then we will set the initial radius to 300nm to emulate the initial dimensions of the pores. With the bulk concentration and the radius fixed, at least within a range, it is observed that only the ratio between c_0 and c_∞, and the value of the line tension τ will change the stability of the system. Furthermore approximate threshold values for these parameters can obtained from the hotspot distance as a function of the diameter, in Fig. (5). Then by assuming a circle-like shape we can establish that the highest unstable mode must fulfill

$$n_{max} > \frac{\pi d}{\langle dz \rangle} \tag{38}$$

With $\langle dz \rangle$ the hotspot distance. By looking at the figure we can obtain the minimum value for the highest mode that will aid with the posterior choice of parameters

$r_0(\mu m)$	n_{max}
0.5	8
1	11
1.5	14
2	16
2.5	18
3	19
3.5	22
4	25

These values are approximated and are only valid for the lineal regime, but they serve as a guideline to choose the range of parameters that we will focus on. We can see from the results that the fist mode will always maintain the same amplitude, this is coherent with the fact that it only corresponds to a translation of the whole interface, and the center of interface will remain in the same place for any sinusoidal mode. It can also be observed that the increase of the radius will

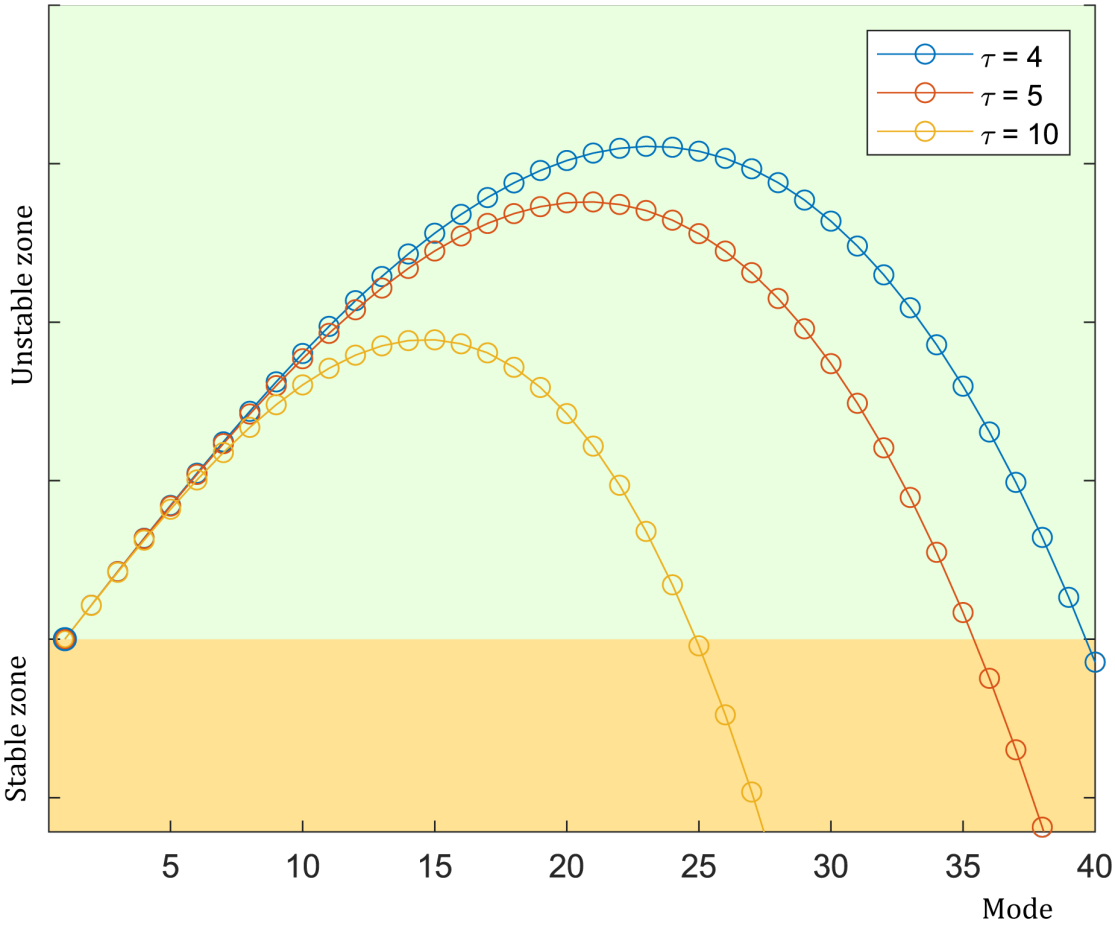

Figure 7: Dispersion relation for different values of τ maintaining the rest of parameters constant. It can clearly be seen that if we increase this parameter the number of unstable frequencies diminishes

increase the unstable modes. This is coherent with the peak splitting instabilities that we want to obtain, since this higher modes will appear as the radius grows.

5 Mathematical derivation of the numerical solution

Once we have a model that it is expected to simulate the binding of the myosin to the membrane and its displacement we need a way to solve it. Because of the irregularity of the pore and the dynamics of the interface it would be impossible to analytically solve this problem, so we will discretize it and use C++to compute the concentration of myosin and evolution of the pores in a 2-D plane.

To do that we will use the Boundary element method. Adapting the derivation encountered in [1] to our case. To use it we start by transforming the partial differential equation into a boundary integral equation. Then the boundary is discretized into constant elements that transform the obtained integral into a linear system that can be easily solved with the desired numerical method. This will provide the concentration at all the points of the domain and boundary and will allow for the computation of the normal derivative of the concentration on the boundary. This derivative is what is needed to obtain the normal velocity of the pore. Then using another integral method we evolve the position of the pore interface. Once this is done the whole process is repeated for every time step to obtain the full evolution of the membrane.

5.1 The Laplace Equation

Since it has been established that the diffusion in the bulk is much faster than the movement of the myosin along the actin filaments we have justified the use of a Laplace equation instead of the Diffusion equation. We start with the definition of the two dimensional Laplace equation

$$\partial_x^2 c(x,y) + \partial_y^2 c(x,y) = 0 \tag{39}$$

Defined in a region $\mathbf{R} \in R^2$, bounded by one or more curves $\mathbf{C}_n$. These curves will define the boundary conditions of the Laplace problem and will provide a unique solution to it.

$$c = f_n(\boldsymbol{p}); \quad \forall \boldsymbol{p} \in \mathbf{C}_n \tag{40}$$

We also define the derivative normal to the curves with the vector $\hat{\boldsymbol{n}} = (n_x, n_y)$ pointing outwards studied region.

$$\partial_{\hat{n}} c = \hat{\boldsymbol{n}} \cdot \boldsymbol{\nabla} c = n_x \partial_x c + n_y \partial_y c \tag{41}$$

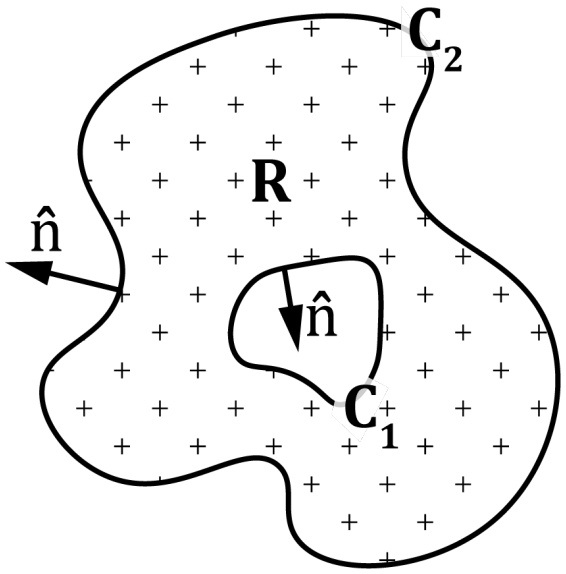

Figure 8: Region of $\mathbf{R}^2$ where the Laplace equation is applied, together with its boundary and the definition of the normal vector.

5.2 Reciprocal relation

In order to apply the boundary element method to the Laplace equation we need to transform the differential equation into the integral form. It can be seen that for two functions $c_1(x, y)$ and $c_2(x, y)$ that fulfill the Laplace equation the following integral can be defined using the Divergence theorem.[1]

$$\int_{\mathbf{C}} \left[c_2 \partial_{\hat{n}} c_1 - c_1 \partial_{\hat{n}} c_2 \right] ds = 0 \tag{42}$$

This integral will allow us for the computation of the function knowing only the values of this function at the boundary $\mathbf{C}$ and a kernel function that we will obtain

later on. From now on we will refer to the reciprocal relation of the functions c_1 and c_2 along the boundary $\mathbf{C}$ using $R_{\mathbf{C}}(c_1, c_2)$

$$R_{\mathbf{C}}(c_1, c_2) = \int_{\mathbf{C}} [c_2 \partial_{\hat{n}} c_1 - c_1 \partial_{\hat{n}} c_2] ds = 0 \tag{43}$$

With this reciprocal relation and the kernel function we can derive the boundary integral equation.

5.3 Kernel function

As it has been said we need a fundamental solution of the Laplace equation to apply the integral boundary equation. In order to do that we will start by assuming that this solution will have polar symmetry, so we will be using polar coordinates.

If we define the nomenclature for the polar coordinates as

$$(x, y) = (r\cos\theta, r\sin\theta) \tag{44}$$

we can rewrite Eq. (39) as

$$\frac{1}{r}\partial_r \left[r\partial_r \phi(r, \theta)\right] + \frac{1}{r^2}\partial_\theta^2 \phi = 0 \tag{45}$$

Then we assume polar symmetry of the solution and we obtain

$$d_r[r d_r \phi(r)] = 0 \quad \forall r \neq 0 \tag{46}$$

This ODE can be solved by integrating twice to obtain

$$\phi(r) = A\ln(r) + B \quad \forall r \neq 0 \tag{47}$$

With A and B arbitrary integration constants that we will set to $\frac{1}{2\pi}$ and 0 respectively for convenience. From this result we can obtain the fundamental solution by changing back to the Cartesian coordinates. We set the origin of the polar coordinates to an arbitrary point $\boldsymbol{p}$ with coordinates (ξ, η) and evaluate the function at the point $\boldsymbol{q} = (x, y)$. The solution will then be valid for any point in the domain

except for the point $p = q$ where its value is undetermined.

$$\Phi(\boldsymbol{q}, \boldsymbol{p}) = \frac{1}{4\pi} \ln \left[(x - \xi)^2 + (y - \eta)^2 \right] = \frac{1}{4\pi} \ln(\boldsymbol{p} - \boldsymbol{q})^2 \qquad \forall \boldsymbol{p} \neq \boldsymbol{q} \tag{48}$$

5.4 Boundary integral solution

Once we have the reciprocal relation as a integral equation and the kernel function we can define a boundary integral equation that will allow us to obtain the values of our desired function in all the domain by evaluating its boundary. We start by taking Eq (48), Eq (43) and taking $c_1 = \Phi$ and $c_2 = c$, which will the concentration at the cytosol defined by Eqs. (39,). This yields

$$\int_{\mathbf{C}} \left[c(\boldsymbol{q}) \partial_n \Phi(\boldsymbol{q}, \boldsymbol{p}) - \Phi(\boldsymbol{q}, \boldsymbol{p}) \partial_n c(\boldsymbol{q}) \right] ds_{\boldsymbol{q}} \qquad \forall \boldsymbol{p} \notin \mathbf{R} \cup \mathbf{C} \tag{49}$$

Since $\Phi(\boldsymbol{q}, \boldsymbol{p})$ is not defined for $\boldsymbol{p} \in \mathbf{R} \cup \mathbf{C}$ because of the undetermined value of $\lim_{x \to 0} \ln |x|$, in order to include this region we replace $\mathbf{C}$ by the union $\mathbf{C} \cup \mathbf{C}_\varepsilon$ with $\mathbf{C}_\varepsilon$ a circle of infinitesimal radius ε and center $\boldsymbol{p}$. Then for $\mathbf{C}$ and $\mathbf{C}_\varepsilon$ we can write the reciprocal relation Eq. (43).

$$\int_{\mathbf{C} \cup \mathbf{C}\varepsilon} [c(\boldsymbol{q}) \partial_n \Phi(\boldsymbol{q}, \boldsymbol{p}) - \Phi(\boldsymbol{q}, \boldsymbol{p}) \partial_n c(\boldsymbol{q})] ds_{\boldsymbol{q}=0} \tag{50}$$

This can be rewritten using the notation defined in Eq. (43) separating them in an integral for every curve as

$$R_{C \cup C_\varepsilon}(\Phi, c) = R_C(\Phi, c) + R_{C_\varepsilon}(\Phi, c) = 0 \tag{51}$$

As long as the circle lies inside R we can make $\varepsilon \to 0$, so we can write

$$R_C(\Phi, c) = - \lim_{\varepsilon \to 0^+} R_{C_\varepsilon}(\Phi, c) \tag{52}$$

From [1] we see that

$$\lim_{\varepsilon \to 0^+} R_{C_\varepsilon}(\Phi, \phi) = -c(\xi, \eta) \tag{53}$$

which means that, using Eq. (51) we obtain

$$c(\boldsymbol{p}) = \int_C [c(\boldsymbol{q})\partial_n \Phi(\boldsymbol{q},\boldsymbol{p}) - \Phi(\boldsymbol{q},\boldsymbol{p})\partial_n c(\boldsymbol{q})] ds_{\boldsymbol{q}} \qquad \forall \boldsymbol{p} \in R \tag{54}$$

Now we need to compute this relation for the boundary $\forall \boldsymbol{p} \in \mathbf{C}$. We take the concentration to be zero outside of the region $\mathbf{R}$ since its contribution is already accounted for in the boundary. Then by repeating the procedure made for $\forall \boldsymbol{p} \in \mathbf{R}$ we take a circle of infinitesimal radius, this time centered on a point of the boundary. Assuming a smooth boundary, for $\varepsilon \to 0$ the boundary becomes a straight line and we only have to integrate for the half of the circle that is inside of the boundary, since the outside part won't contribute to the integral. This yields

$$\int_{\mathbf{C}_\varepsilon} c(\boldsymbol{p})\partial_n \Phi(\boldsymbol{q},\boldsymbol{p}) ds = -\frac{1}{2} c(\boldsymbol{p}) \qquad \forall \boldsymbol{p} \in \mathbf{C} \tag{55}$$

Which means that Eq. (51) will become

$$\frac{1}{2} c(\boldsymbol{p}) = \int_{\mathbf{C}} [c(\boldsymbol{q})\partial_n \Phi(\boldsymbol{q},\boldsymbol{p}) - \Phi(\boldsymbol{q},\boldsymbol{p})\partial_n c(\boldsymbol{q})] ds_{\boldsymbol{q}} \qquad \forall \boldsymbol{p} \in \mathbf{C} \tag{56}$$

After this steps have the concentration at any point of the studied region and its boundary

$$(\boldsymbol{p})c(\boldsymbol{n}) = R_C(\Phi, c) = \int_{\mathbf{C}} [c\partial_{\hat{n}}\Phi - \Phi\partial_{\hat{n}} c] ds \qquad \forall \boldsymbol{p} \in \mathbf{R} \tag{57}$$

$$\frac{1}{2}(\boldsymbol{p})c(\boldsymbol{n}) = R_C(\Phi, c) = \int_{\mathbf{C}} [c\partial_{\hat{n}}\Phi - \Phi\partial_{\hat{n}} c] ds \qquad \forall \boldsymbol{p} \in \mathbf{C} \tag{58}$$

This is the full integral solution for the Laplace equation.

5.5 Boundary solution with constant elements

The boundary can be approximated by a N-sided polygon with sides C^i such that

$$\mathbf{C} \approx \bigcup_{i=1}^{N} C^i \tag{59}$$

Where we put N well spaced points $\boldsymbol{p}^i \in \mathbf{C}$ and the element C^i will be the segment

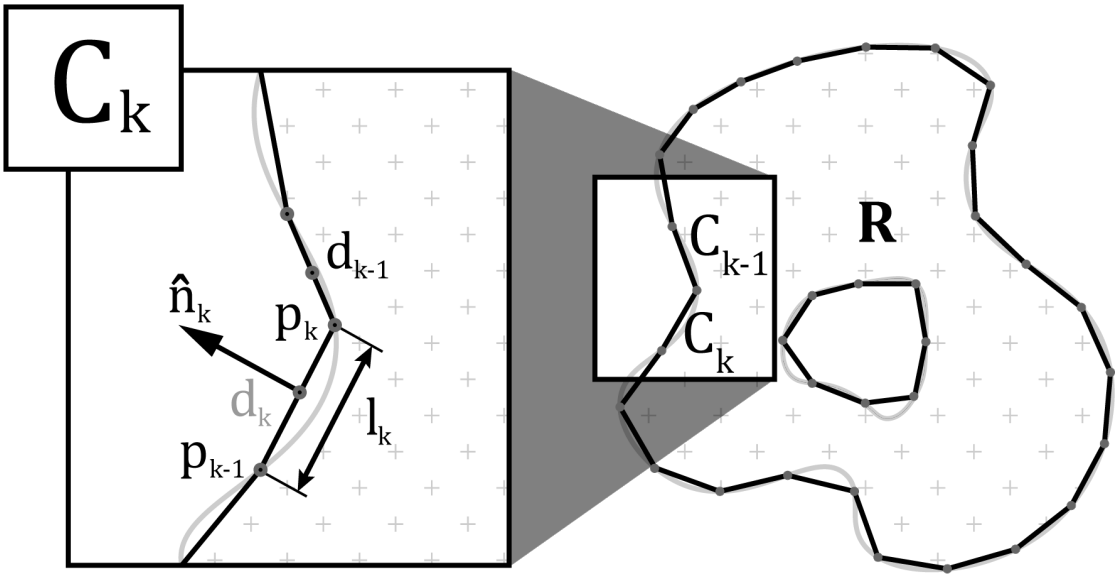

Figure 9: Discretization of the boundary of the studied region.

that goes from $\boldsymbol{p}^i$ to $\boldsymbol{p}^{i+1}$. In this approximation is assumed that ϕ and $\partial_n \phi$ are constant on each element, taking the value of the function at the midpoint of the segment.

$$c^i = c(d^i) \qquad v^i = \partial_{n^i} c(d^i) \tag{60}$$

$$d^i = \frac{\boldsymbol{p}^i + \boldsymbol{p}^{i+1}}{2} \tag{61}$$

By using this approximations we can rewrite [57] as

$$c(\boldsymbol{p}) = \sum_{k=1}^{N} [c^k I_2^k(\boldsymbol{p}) - v^k I_1^k(\boldsymbol{p})] \qquad \forall p \in \mathbf{R} \tag{62}$$

$$\frac{1}{2} c(\boldsymbol{p}) = \sum_{k=1}^{N} [c^k I_2^k(\boldsymbol{p}) - v^k I_1^k(\boldsymbol{p})] \qquad \forall p \in \mathbf{R} \tag{63}$$

$$I_1^k(\boldsymbol{p}) = \int_{C^k} \Phi(\boldsymbol{q}, \boldsymbol{p}) ds_{\boldsymbol{q}} \tag{64}$$

$$I_2^k(\boldsymbol{p}) = \int_{C^k} \partial_{n^k} \Phi(\boldsymbol{q}, \boldsymbol{p}) ds_{\boldsymbol{q}} \tag{65}$$

Then we can determine the values of the normal derivative of the function at the boundary by evaluating [62] at the midpoints of the segments.

$$\frac{1}{2} c^m = \sum_{k=1}^{N} [c^k I_2^k(\boldsymbol{d^m}) - v^k I_1^k(\boldsymbol{d^m})] \tag{66}$$

In order to obtain the values we will take $m = 1, ..., N$ and obtain the following linear system

$$\mathbf{M_1} \cdot \boldsymbol{v} = \mathbf{M_2} \cdot \boldsymbol{c} \tag{67}$$

$$\mathbf{M_1}_k^m = I_1^k(d^m) \tag{68}$$

$$\mathbf{M_2}_k^m = I_2^k(d^m) - \frac{1}{2} \delta_k^m \tag{69}$$

With δ_k^m Kronecker delta. Then if we solve for $\boldsymbol{v}$ we obtain the normal derivative at the boundary that will allow us to obtain the normal velocity of the interface

$$\boldsymbol{v} = \mathbf{M_1}^{-1} \mathbf{M_2} \boldsymbol{c} \tag{70}$$

Once we know both c^k and v^k we can use Eq. (62) with $\lambda = 1$ to compute the values $\forall \boldsymbol{p} \in \mathbf{R}$

$$c(\boldsymbol{p}) \approx \sum_{k=1}^{N} [c^k I_2^k(\boldsymbol{p}) - v^k I_1^k(\boldsymbol{p})] \tag{71}$$

5.6 Evolution of the curve

Variables are labeled with a subindex i if they are evaluated at the beginning of the segment, at $\boldsymbol{p_i}$, and a subindex j if they are evaluated in the middle point, d_i. To convert from a function evaluated at the beginning of the segments f_i to a function evaluated at the midpoints f_j we use

$$f_j = \frac{f_i + f_{i+1}}{2} \qquad f_i = \frac{f_{j-1} + f_j}{2} \tag{72}$$

With this nomenclature, the variables that we obtain from the boundary element method are c_j^t and v_j^t. This values provide enough information evolve the curve for one time step.

The evolution of the curve is given only by the normal velocity of the curve, as exposed in [6]. To evolve the curve we will use the the integral method from the aforementioned paper. We can compute the evolution of the perimeter by using the following integral

$$\partial_t L = \int_0^1 v_n \partial_\rho \theta d\rho \tag{73}$$

With ρ an arc lent parametrization escalated to range from 0 to 1 along the curve. Now we need to define a discrete integration method. Since the evaluation of the functions is in the middle points the numerical integration will be trapezoidal Then, any integral will become

$$\int_a^b f(t)dt \approx Int_t \left[f_i\right]_a^b \tag{74}$$

With

$$Int_t[f_i]_a^b = \sum_{i=0}^{N-1} \frac{(f_i + f_{i+1})(t_{i+1} - t_i)}{2} = \sum_{i=0}^{N-1} \frac{f_j(t_{i+1} - t_i)}{2} \tag{75}$$

For a vector with N samples. If we use the results obtained from solving the PDE the discretized derivative of the perimeter becomes

$$d_t L^t = Int_\rho [v_i^t \partial_\rho \theta_i^t]_0^1 \tag{76}$$

Then L_t, the discretized version of $L(t)$, is obtained by using a simple Euler method

$$L_t = L_{t-\Delta t} + \Delta t \cdot d_t L_{t-\Delta t} \tag{77}$$

Once we have the updated length we can compute the evolution of the angle of the segments of the interface with the positive horizontal axis $\theta(\rho)$. First we obtain the partial derivative $\partial_t \theta(t, \rho)$ given by

$$\begin{aligned}\partial_t \theta(t,\rho) = &-\frac{1}{L(t)}\partial_\rho V_n(t,\rho) \\ &- \frac{1}{L(t)}\partial_\rho \theta \left(\rho \int_0^1 V_n(t,\rho)\partial_\rho \theta d\rho - \int_0^\rho V_n(t,\rho)\partial_\rho \theta d\rho\right)\end{aligned} \tag{78}$$

In the discretized notation will become

$$d_t\theta_i^t = -\frac{1}{L^t} d_\rho v_{ni}^t - \frac{1}{L^t} d_\rho \theta_i \left[\rho_i Int_\rho[v_{ni}^t d_\rho \theta_i^t]_0^1 - Int_\rho[v_{ni}^t d_\rho \theta_i^t]_0^{\rho_i}\right] \tag{79}$$

Then we evolve the discretized θ_i^t using again the Euler integration.

$$\theta_i^t = \theta_i^{t-\Delta t} + \Delta t d_t \theta_i^{t-\Delta t} \tag{80}$$

Once we have the updated theta we can retrieve the values of the positions of the interface by using the following integrals

$$x(t,\rho) = x(0,0) + \int_0^t v_n(t',0)\sin\theta(t',0)dt' + L(t)\int_0^\rho \cos\theta(t,\rho')d\rho' \tag{81}$$

$$y(t,\rho) = y(0,0) - \int_0^t v_n(t',0)\cos\theta(t',0)dt' + L(t)\int_0^\rho \sin\theta(t,\rho')d\rho' \tag{82}$$

Which will be discretized in the following way

$$x_i^t = x_0^0 + Int_t[v_{n0}^{t'}\sin\theta_0^{t'}]_0^t + L_t Int_\rho[\cos\theta_i^t]_0^\rho \tag{83}$$

$$y_i^t = y_0^0 - Int_t[v_{n0}^{t'}\cos\theta_0^{t'}]_0^t + L_t Int_\rho[\sin\theta_i^t]_0^\rho \tag{84}$$

This leads to the new position of the interface which can be used to solve again the Laplace equation and repeat the process for the next time step.

6 Computer implementation

With the discretized model fully developed we start to implement it to a computer. To do that we need to choose a computer language to convert it. After some consideration C++ was chosen for the computation of the evolution of the interface for its speed and versatility.

From the mathematical derivation it is clear that a library for the creation and arithmetics will be needed. In order to supply that, the Eigen library was used. With this library the creation and algebraic manipulation of matrices were much more straightforward. Since we are working with an infinite two-dimensional bulk

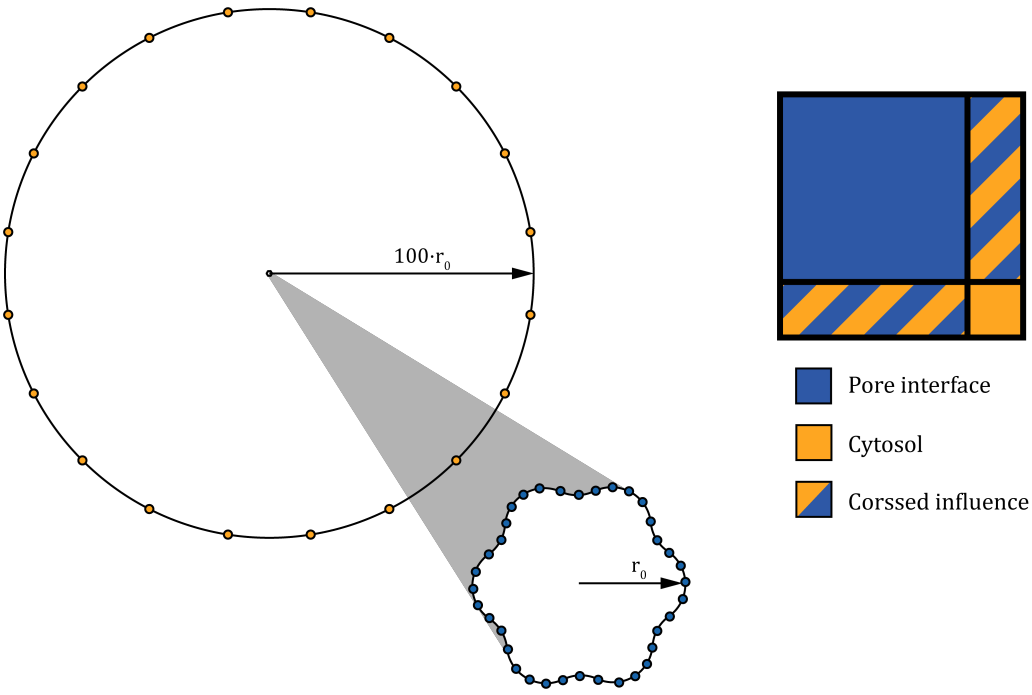

Figure 10: Points of the discretitzed boundary where the concentration is known. This includes both the interface boundary and points that set the concentration of the solution far from the interface. Representation of the space that these points fill in the Boundary element matrices, this matrices compute the influence of each point over all the others. Only the yellow part is constant. The blue and mixed color parts will change as the interface evolves since the distance between the points will change, meaning that the influence they exert to each other will also be modified.

we need to bound it at a particular radius in order to set the points where we set the concentration of the bulk. This points will be set in along a circumference of radius a hundred times bigger than r_0, as seen in the linear stability analysis. Then the collocation points will be as shown in Fig [10]. To solve the system of linear equations created by the boundary element method, we chose the Householder QR decomposition algorithm, provided by the Eigen library, which offered the best trade-off between velocity and accuracy. And a fast matrix solver is an important aspect for this particular problem. This is because for our moving boundary problem almost all of the matrix needed to be recalculated at every time step, as seen in Fig. [10]. This means that if the matrix solving algorithm is not efficient the time of the simulation would be impractically high and the number of points used to discretize the boundary would need to be reduced. This carries with it problems since the distance between the points of the boundary has to be small in order to avoid numeric instabilities. This problem also gets aggravated by the fact that the number of points to which the interface is discretized also grows with the pore perimeter. This is done to maintain the distance between consecutive points bounded within the numerically stable range. To do this every time the distance between two consecutive points is greater than a fixed threshold we interpolate the the known positions of the interface to double the number of points.

Taking this limitations into account we wrote the code that would solve and evolve the model. And to test it we started by trying the growth of single mode interfaces. In order to keep all the points equispaced in the interface and minimize the numerical instabilities we set the initial condition analogously to [[6]] and set the initial condition for the angle of each point to

$$\theta_n(\rho) = 2\pi\rho + \delta \sin(2\pi n\rho) \tag{85}$$

With n the mode and ρ the parametrization of the curve, ranging from zero to one. Then scale the points to the initial radius and run and start advancing time. This will help choose the remaining parameters in order to obtain a growth similar to the one seen in the experiments.

Once the model is tested we set different initial conditions, we started with a sum of modes to see the competition between them.

$$\theta(\rho) = 2\pi\rho + \sum_{n=1}^{N} \delta(n) \sin(2\pi n\rho + u(n)) \tag{86}$$

Where we create an initial interface that contains a sum of the first the first N modes, with a random phase induced by the uniform distribution $u(n)$, that goes from 0 to π. These modes will vanish or increase and interact between them, giving . In the next section we will review the results of these simulations. Finally we also used a circle perturbed by a random Gaussian noise that would better represent the irregularities in the membrane that would act as seed for the growth of the pore. in the form of

$$\theta(\rho) = 2\pi\rho + g_\delta(\rho) \tag{87}$$

With g_δ a random number extracted from a Gaussian distribution centered at zero and with a standard deviation of δ. By repetition of the simulations and analysis and comparison of the results with the experiments a set of parameters was chosen. Some of them were given by the experimental results and the other were obtained from fitting the results of the simulations.

7 Results and discussion

Once the parameters of the model are chosen we can start to extract results from the simulations. These will be the variation in the growth with the concentration and the evolution of the pore to, taking special attention to the splitting of the hotspots with the growth of the radius due to the growth of the unstable region seen in the linear stability analysis.

7.1 Concentration dependency

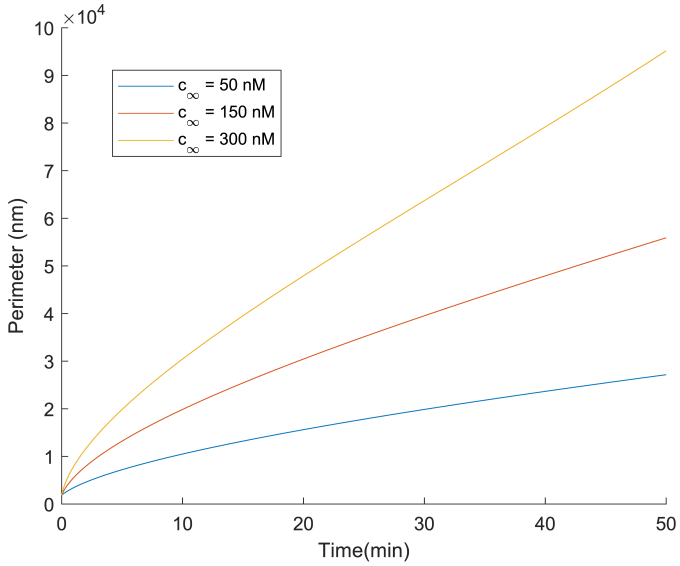

Figure 11: Evolution of the perimeter for different concentrations, it can be seen that the velocity of the interfaces grows with the concentration without changing its profile

One of the effects that couldn't be obtained from the phenomenological model in [6] was the dependency of the growth of the pores with the concentration. The experimental data clearly depicted an increase of the velocity of growth with the increase of myosin VI in the cytosol. One of the main objectives of this work was to obtain a dependency with the concentration that resembled the one seen in the experiments. To do this we created circular pores, which needed less computational time to grow since they do not need many points for the solutions to be numerically stable.

This yields the results shown in Fig. (11), that can be compared to the experimental ones in Fig. (5). We observe that the changes in concentration create a change in the timescale of the whole problem, augmenting the velocity of the process for higher concentrations. Through this comparison the parameters such as the relation between c_0 and c_∞, and c_i could be further polished to better fit the experimental results available.

7.2 Pore growth

Since this is a non-local model the whole interface contributes to the evolution of every point of the interface. This hinders the fitting of the parameters since for different initial seeds the evolution of the interface can vary significantly.

In order to obtain these parameters, the ones that are not fixed, like the concentration or the initial radius, several initial conditions have been tested. The parameters for each of those where chosen according to the linear stability analysis preformed.

Each seed was initialized with a slightly different set of parameters, shown below,that allowed us to observe different behaviours.

Seed number	1	2	3	4	5	6
c_0	0.5	0.5	1	1	0.5	0.5
c_∞	50	50	50	50	50	50
$\frac{D}{c_0}$	17	17	17	17	17	17
r_0	200	200	200	300	200	200
τ	10	4	10	5	4	4.5

Then these results where compared with the with the experiments in order to see if they could accurately model them. It can be seen, Fig (13), that the radius versus perimeter power law is fulfilled in all the cases. The distance between hotspots fluctuates more aggressively than the experimental data. This may be because the exactly correct seed to start the simulation hasn't been found. On the other hand is clear that seeds with large curvature changes, like the second and the sixth create a higher first spike. Nevertheless the overall trend seems to be stabilizing at a fixed value even with the initial instabilities. In this plot the fifth seed wasn't

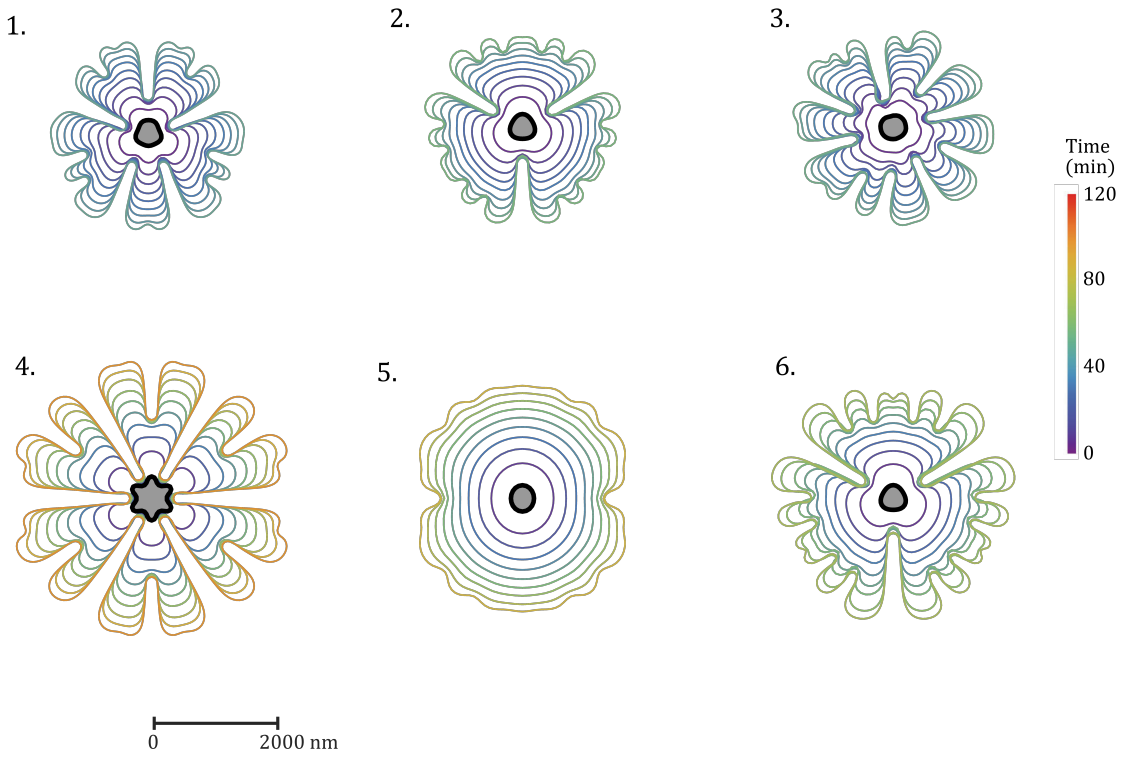

Figure 12: Results for the evolution of the pore. In the figure it can be seen the importance of the initial conditions in the growth of the pore. The seeds with higher mode perturbations such as 4. will led to more dramatic inward curvature. And the seeds with lesser modes, such as 5. will no provide instabilities to accurately model growth of the pore. 2. and 6. have the same initial condition but the increased τ in 6 will make it harder for the instabilities to appear. Finally we have 1. the sum of different modes and 3. with a seed that is perturbed by a random Gaussian noise. The irregularity of the noise breaks the symmetry seen in the other flowers.

included. Finally for the concentration we can see that in general the perimeter of the pores grows at the expected rate. The only one that seems to deviate more from this trend is the fifth seed, whose oval shape kept the the perimeter from growing by avoiding the inward curvature and peak splitting present in the other pores.

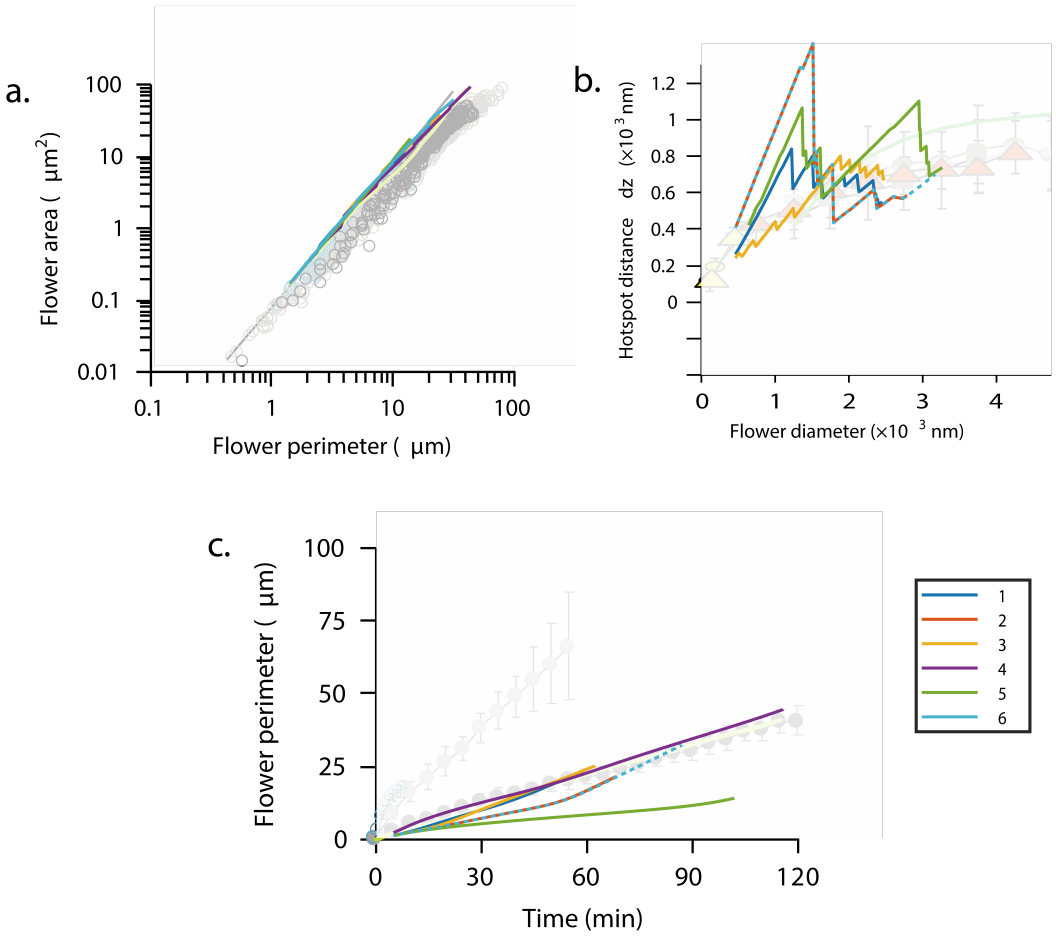

Figure 13: Results of the simulation for the six different initial conditions compared against the experimental data. The plots of the experimental data have been cleaned to avoid clumping. To see the whole information refer to Fig. (5).

8 Conclusions and further work

In this bachelor's thesis a model for the binding of myosin to the membrane has been developed and numerically solved, focusing on the importance of the curvature of the interface in the cooperative aspect of this binding.

The model for this phenomenon has been proved to provide results that evolve accordingly to the pore growth seen in the experiments. This results both broadly agree with he concentration changes and the hotspot distance. More simulations should be done to further test the model and try to find other possible pore shapes by changing the initial conditions.

Although it agrees with the data this model is a simple one and much more factors should be taken into consideration in order to fully grasp this interaction. One of the first upgrades to the model would be adding the diffusion therm that we neglected once we were deriving it. It was deemed negligible due to the value of the Péclet number relative to this case,

$$Pe = \frac{v_n r_0}{D} \sim 10^{-2} \tag{88}$$

This is true for the values of concentration used in the experimental data available, where the timescale of growth of the pores was two orders of magnitude greater than the diffusion timescale. Although this approximation makes sense with the studied concentrations, we have seen that the change in the concentration of the myosin greatly affects the timescale of growth, so for a high enough concentration we would need to take into account the diffusion in the bulk. This would affect the stability of the whole system and maybe generate new emergent phenomena in the pore growth.

There are more biological aspects that could be taken into account such as the rate of unbinding between the myosin and the interface, since it eventually breaks its bound and diffuses back to the cytosol. This effect was also considered when designing the model but since it would not be significant for the timescales we are working with it is not included.

The model could also be enhanced in the computational front. In this work the

boundary has been discretized by using a certain number of constant elements that transform the smooth curve into a polygon. Then each element of the polygon is considered to have a constant value of concentration of myosin. This could be enhanced by allowing each element to have a linear change of concentration, instead of being constant. This method would lead to improved accuracy of the results, but it would also have the trade off of having to double the size of the matrices used. As it has been discussed in the computational part the matrix size is a key aspect to this model, because its moving boundary requires for a recalculation of almost all of the coefficients every time step. So in the conditions this work was made doubling the size off the matrices would have led to unpractical computation times. And further work would need to be done to see if its better to discretize the boundary into a higher number of constant elements or to keep the number of elements constant and instead make its concentration along them linear.

Finally this work could be preformed into a full two-dimensional membrane capable of bending and acquire different shapes instead of one-dimensional interface in order to replicate more complex processes such as budding or cell motility and work with a three-dimensional bulk. .

9 Appendix: Code

9.1 Matlab

```
clear variables
%% Initial config

N1 = 20;
N2 = 20;
N = N1 + N2;

p = zeros(N,2);
tau = 4.5;
r0 = 300;
r1 = 100*r0;
c1 = 300;
c0 = 0.1*c1;
k1 = 00000000;
n = zeros(N,2);
dp = 1;
thi = zeros(N1,1);
rhoi = linspace(0,1,N1+1);
for ii = 2:N1
    thi(N1-ii+1) = 2*pi*rhoi(ii) + normrnd(0,dp);
end
thi = circshift(thi,1);
thi(1) = 2*pi;
thj = 0.5.*(thi + circshift(thi,-1));
thj(end) = thj(end)-pi;
for ii = 2:N1
    p(ii,:) = p(ii-1,:) + [cos(thj(ii-1)),sin(thj(ii-1))];
end
p = [p(:,1)-mean(p(1:N1,1)) p(:,2)-mean(p(1:N1,2))];
```

```
r = sqrt(p(1:N1,1).^2 + p(1:N1,2).^2);
p = p./mean(r)*r0;
p(N1+1:end,:) = r1*[cos(2*pi/N2*[1:N2])' sin(2*pi/N2*[1:N2])'];
ttf = 10000 ; tt = 1;
p0 = p(1:N1,:);
l = zeros(N,1); cm = zeros(N,2);
stg = 0;
stgp = cell(ttf/1000,1);
lt = 2*pi*r0/N1*1000;
F1 = zeros(N); F2 = F1; M2 = F1; M1 = F1;
phi = zeros(N,1); v = zeros(N,1);
aa = 1; d = 1; dt = 0.5;
init = 1;
pn = zeros(N1,2);
%%
while tt < ttf
    for ii = 1:N1
        cm(ii,:) = (p(ii,:) + p(ii+1,:))./2;
        l(ii) = sqrt((p(ii+1,1) - p(ii,1))^2 + (p(ii+1,2) - p(ii,2))^2);
        n(ii,:) = [(p(ii+1,2)-p(ii,2)) (p(ii,1)-p(ii+1,1))]./l(ii);
        if ii == N1
            cm(ii,:) = (p(ii,:) + p(1,:))./2;
            l(ii) = sqrt((p(1,1) - p(ii,1))^2 + (p(1,2) - p(ii,2))^2);
            n(ii,:) = [(p(1,2)-p(ii,2)) (p(ii,1)-p(1,1))]./l(ii);
        end
    end
    for ii = 1:N2
        if ii == N2
            cm(ii + N1,:) = (p(ii + N1,:) + p(N1 + 1,:))./2;
            l(ii + N1) = sqrt((p(1+N1,1) - p(ii+N1,1))^2 +
            ...(p(1+N1,2) - p(ii+N1,2))^2);
            n(ii + N1,:) = [(p(1+N1,2)-p(ii+N1,2))
```

```
        ...(p(ii+N1,1)-p(1+N1,1))]./l(ii+N1);
    else
        cm(ii + N1,:) = (p(ii + N1,:) + p(ii + N1 + 1,:))./2;
        l(ii+N1) = sqrt((p(ii+1+N1,1) - p(ii+N1,1))^2 + (p(ii+1+N1,2)
        ...- p(ii+N1,2))^2);
        n(ii+N1,:) = [(p(ii+1+N1,2)-p(ii+N1,2)) (p(ii+N1,1)-
        ...p(ii+1+N1,1))]./l(ii+N1);
    end
end
L = sum(l(1:N1));
if max(l(1:N1))> lt
    N1 = 2*N1;
    drhoj = 1/N1*ones(N1,1);
    N = N1 + N2;
    p = zeros(N,2);
    ppn = pn(1,:);
    pn = zeros(N1,2);
    pn(1,:) = ppn;
    thin = zeros(N1,1);
    for ii = 1:N1/2
        thin(2*ii-1) = thi(ii);
        thin(2*ii) = thj(ii);
    end
    thj = 0.5.*(thin + circshift(thin,-1));
    thj(end) = thj(end)-pi;

    for ii = 2:N1
        pn(ii,:) = pn(ii-1,:) + L*[cos(thj(ii-1))*drhoj(ii-1)
        ...sin(thj(ii-1))*drhoj(ii-1)];
    end
    thi = thin;
    p(1:N1,:) = pn;
```

```
    p(N1+1:end,:) = r1*[cos(2*pi/N2*[1:N2])' sin(2*pi/N2*[1:N2])'];
    F1 = zeros(N); F2 = F1; M2 = F1; M1 = F1; l = zeros(N,1); cm = zeros(N,2);
    phi = zeros(N,1); v = zeros(N,1);
    init = 1;
else
    dthj = zeros(N1,1);
    for ii = 1:N1-1
        dthj(ii) = -(thi(ii+1) - thi(ii))/l(ii)*L;
    end
    dthj(N1) = -(thi(1) - thi(N1)-2*pi)./l(N1)*L;
    bc = zeros(N,1);
    for ii = 1:N1
        bc(ii) = c0*(1+tau*((dthj(ii)/L)));
    end
    for ii = 1:N2
        bc(ii+N1) = c1;
    end
    % Matrix definition

    for mm = 1:N
        for kk = 1:N
            if init == 0 && mm>N1 && kk >N1
            else
                A = l(kk)*l(kk);
                B = 2*l(kk)*(-n(kk,2)*(p(kk,1)-
                ...cm(mm,1))+n(kk,1)*(p(kk,2)-cm(mm,2)));
                E = (p(kk,1)-cm(mm,1))^2+(p(kk,2)-cm(mm,2))^2;
                ff3 = 4*E*A-B^2;
                if ff3 < 1e-11
                    ff1 = B/(2*A);
                    F1(mm,kk) = l(kk)/(2*pi)*(log(l(kk))+
                   ... (1+ff1)*log(abs(1+ff1))-ff1*log(abs(ff1))-1);
```

```
                F2(mm,kk) = 0;
            else
                ff1 = B/A; ff2 = E/A; ff3 = sqrt(abs(ff3));
                f21 = l(kk)*(n(kk,1)*(p(kk,1)-cm(mm,1))
               ... + n(kk,2)*(p(kk,2)-cm(mm,2)))/(pi*ff3);
                f22 = atan((2*A+B)/ff3)-atan(B/ff3);
                F2(mm,kk) = f21*f22;
                F1(mm,kk) = l(kk)/(4*pi)*(2*(log(l(kk))-1)-
                ...ff1/2*log(abs(ff2))+(1+ff1/2)
                ...*log(abs(1+ff1+ff2))+ff3/A*f22);
            end
        end
    end
end
% BEM solve
for mm = 1:N
    for kk = 1:N
        M1(mm,kk) = F1(mm,kk);
        M2(mm,kk) = F2(mm,kk) - 0.5*eq(mm,kk);
    end
end
b = M2*bc;
c = M1\b;

for ii = 1:N
        phi(ii) = bc(ii);
        v(ii) = c(ii);
end
% Vel


dL = zeros(N1+1,1);
```

```
vnj = -aa^2*d*v(1:N1);
drhoj = l(1:N1)/L;
dkj = (circshift((dthj/L),-1) + circshift((dthj/L),1)
...-2*(dthj/L))./(l(1:N1).^2);
dkj = 0.25*(2*dkj + circshift(dkj,1) + circshift(dkj,-1));
vnj = vnj + k1*dkj;
vni =  0.5.*(vnj + circshift(vnj,1));
dvnj = (circshift(vni,-1)-vni)./drhoj(1:N1);

rhoi = [0;cumsum(l(1:N1))/L];
rhoj = 0.5.*(rhoi + circshift(rhoi,-1));
rhoj = rhoj(1:N1);

dL(1) = 0;
for ii = 2:N1+1
    dL(ii) = dL(ii-1) + drhoj(ii-1)*dthj(ii-1)*vnj(ii-1);
end
dLj = 0.5.*(dL + circshift(dL,-1)); dLj = dLj(1:N1);

ff = rhoj.*dL(end) - dLj;
dtthj = 1/L*dvnj -1/L*dthj.*ff;
thj = thj + dt*circshift(dtthj,0);
thi = 0.5*(thj + circshift(thj,1));
thi(1) = thi(1) + pi;
ll(tt) = L;
pn(1,:) = [(p(1,1) - dt*vni(1)*sin(thi(1)))
...(p(1,2) + dt*vni(1)*cos(thi(1)))];
L = L + dt*dL(end);
for ii = 2:N1
    pn(ii,:) = pn(ii-1,:) + L*[cos(thj(ii-1))*drhoj(ii-1)
    ...sin(thj(ii-1))*drhoj(ii-1)];
end
```

```
        p(1:N1,:) = pn;
        if mod(tt,1000) == 0
            stg = stg +1;
            stgthi{stg} = 0.5*(dthj + circshift(dthj,-1));
            stgp2{stg} = pn;
        end
        tt = tt + 1;
        ll(tt) = L;
        init = 0;
    end
end
}
```

9.2 C++

```
#include <iostream>
#include <fstream>
#include <Eigen/Dense>
#include <math.h>
#include <random>
using namespace std;
using Eigen::MatrixXd;
using Eigen::ArrayXXd;
using Eigen::ArrayXd;
using Eigen::VectorXd;


//////////////////
// Declarations //
//////////////////

// Parameter definition
```

```
const int N1 = 500; // Number of elements in the interface
const int N2 = 50; // Number of elements in the bulk boundary
const int N = N1 + N2; // Total number of points
const long double r1 = 216; // Initial radius of the interface
const long double r2 = r1*100; // Radius of the bulk points
const long double tau = -1; // Line tension
const long double c0 = 0.1; // Bulk concentration
int ttf = 10000; // Steps of the simulation
long double dt = 0.0001; // Time step
const long double tf = dt*ttf; // Time of simulation
const long double stddev = 0.0;
long double dp = 0.05; //Amplitude of the perturbation
long double Ln,L,dL,ff;
int ttstg = 0; // Number of times that the perimeter is saved
int tt = 0; // Current iteration number
int tt0 = 0; //
bool init = 1;// If saved initial conditions init = 0, else init = 1
long double xx0, yy0;
const long double a = 10,d = -1;
double rnd,md;
int mdmax = 20;
double xin,yin;
default_random_engine gen;
normal_distribution<long double> noise(r1,stddev);


// Variable declaration

ArrayXXd p(N,2), n(N,2),cm(N,2);
ArrayXd lj(N),vnj(N1), dvnj(N1), vni(N1), rhoj(N1),
...rhoi(N1), thj(N1), dthi(N1),
...dthj(N1), unif(ttf+N);
```

```
ArrayXd dtthj(N1), dtthi(N1), thi(N1), thin(N1), mmj(N1);
ArrayXd ll(ttf+1),vnit(ttf),thit(ttf), mdr(mdmax);
MatrixXd F1(N,N), F2(N,N);
VectorXd cj(N), dcj(N), mm(N);



///////////////
// Functions //
///////////////

// Basic integration function

double integ(int N, ArrayXd lj,ArrayXd fi) {
  double in = 0;
  for (int ii = 0; ii < N-1; ii++) {
    in += 0.5*lj(ii)*(fi(ii) + fi(ii+1));
  }
  return in;
}

double integj(int N, ArrayXd lj,ArrayXd fj) {
  double in = 0;
  for (int ii = 0; ii < N; ii++) {
    in += lj(ii)*fj(ii);
  }
  return in;
}

// Basic cosine integration function

double cinteg(int N, ArrayXd lj,ArrayXd fi) {
```

```
  double in = 0;
  for (int ii = 0; ii < N-1; ii++) {
    in += lj(ii)*cos((fi(ii) + fi(ii+1))/2);
  }
  return in;
}

// Basic cosine integration function

double sinteg(int N, ArrayXd lj,ArrayXd fi) {
  double in = 0;
  for (int ii = 0; ii < N-1; ii++) {
    in += lj(ii)*sin((fi(ii) + fi(ii+1))/2);
  }
  return in;
}

// From point to midpoint
ArrayXd itoj(int N, ArrayXd fi) {
  ArrayXd fj(N);
  for (int ii = 0; ii < N-1; ii++) {
    fj(ii) = 0.5*(fi(ii) + fi(ii+1));
  }

  fj(N-1) = 0.5*(fi(0) + fi(N-1));
  return fj;
}

// From midpoint to point

ArrayXd jtoi(int N, ArrayXd fj) {
  ArrayXd fi(N);
```

```
  for (int ii = 1; ii < N; ii++) {
    fi(ii) = 0.5*(fj(ii) + fj(ii-1));
  }
  fi(0) = 0.5*(fj(0) + fj(N-1));
  return fi;
}

// Unwraping function for the angular variables

ArrayXd unwrap(int N, ArrayXd f) {
  bool wrap = 1;
  while (f(0) > 6) {
    f(0) -= 2*M_PI;
  }
  while (f(0) < 3) {
    f(0) += 2*M_PI;
  }
  while (wrap == 1) {
    wrap = 0;
    for (int ii = 1; ii < N; ii++) {
      if (f(ii)-f(ii-1) > 4) {
        f(ii) -= 2*M_PI;
        wrap = 1;
      } else if (f(ii)-f(ii-1) < -4) {
        f(ii) += 2*M_PI;
        wrap = 1;
      }
    }
  }
  return f;
}
```

```
/////////////////
// Main script //
/////////////////


int main() {
  // Openig files for saving data
  ofstream stgdata;
  if (init == 1) {
    stgdata.open("stgdata.txt",ofstream::out);
  } else {
    stgdata.open("stgdata.txt",ofstream::out | ofstream::app);
  }


  // Initial conditions
  if (init == 1) {
    for (int jj = 0; jj < mdmax; jj++){
      mdr(jj) = 100*noise(gen);
    }
    for (int ii = 1; ii<N1 + 1; ii++) {
      // Circle + noise
      rnd = noise(gen);
      p.row(N1-ii) << rnd*cos(2*ii*M_PI/N1),rnd*sin(2*ii*M_PI/N1);

      //Circle
      //p.row(N1-ii) << r1*cos(2*ii*M_PI/N1),r1*sin(2*ii*M_PI/N1);

      //Circle + mode
      p.row(N1-ii) << r1*(1+dp*cos(8*ii*M_PI/N1))*cos(2*ii*M_PI/N1),r1*
      ...(1+dp*cos(8*ii*M_PI/N1))*sin(2*ii*M_PI/N1);
```

```
    //Circle + modes
    md = 0;
    for (int jj = 1; jj <mdmax; jj++){
      md += dp*cos(2*jj*ii*M_PI/N1);
    }
    //p.row(N1-ii) << rnd*(1+md)*cos(2*ii*M_PI/N1),rnd*(1+md)*sin(2*ii*M_PI/N1);
  }
  for (int ii = 1; ii<N2 + 1; ii++) {
    p.row(N1 + ii - 1) << r2*cos(2*ii*M_PI/N2),r2*sin(2*ii*M_PI/N2);
  }
} else {
  cin >> tt0;
  for (int ii = 0; ii < N ; ii++) {
    cin >> xin >> yin;
    p.row(ii) << xin,yin;
  }
}
xx0 = p(0,0);
yy0 = p(0,1);

for (int ii = 0; ii<ttf+N; ii++) {
  unif(ii) = 1;
}
// Time evolution

while (tt < ttf) {
  for (int ii = 0; ii<N1-1; ii++) {
    lj(ii) = sqrt(pow(p(ii+1,0)-p(ii,0),2) + pow(p(ii+1,1)-p(ii,1),2));
    n.row(ii) << (p((ii+1),1)-p(ii,1))/lj(ii),(p(ii,0)-p((ii+1),0))/lj(ii);
    cm.row(ii) << (p(ii,0) + p(ii+1,0))/2,(p(ii,1) + p(ii+1,1))/2;
  }
  lj(N1-1) = sqrt(pow(p(0,0)-p(N1-1,0),2) + pow(p(0,1)-p(N1-1,1),2));
```

```
n.row(N1-1) << (p(0,1)-p(N1-1,1))/lj(N1-1),(p(N1-1,0)-p(0,0))/lj(N1-1);
cm.row(N1-1) << (p(N1-1,0) + p(0,0))/2,(p(N1-1,1) + p(0,1))/2;

for (int ii = N1; ii<N-1; ii++) {
  lj(ii) = sqrt(pow(p(ii+1,0)-p(ii,0),2) + pow(p(ii+1,1)-p(ii,1),2));
  n.row(ii) << (p((ii+1),1)-p(ii,1))/lj(ii),(p(ii,0)-p((ii+1),0))/lj(ii);
  cm.row(ii) << p.row(ii)/2 + p.row(ii + 1)/2;
}
lj(N-1) = sqrt(pow(p(N1,0)-p(N-1,0),2) + pow(p(N1,1)-p(N-1,1),2));
n.row(N-1) << (p(N1,1)-p(N-1,1))/lj(N-1),(p(N-1,0)-p(N1,0))/lj(N-1);
cm.row(N-1) << p.row(N-1)/2 + p.row(N1)/2;

// Curvature of the interface


if(tt == 0){
  L = integ(N1+1,lj,unif);
  for (int ii = 0; ii<N1; ii++) {
    if (p(ii+1,1) - p(ii,1) > 0){
      if (p(ii+1,0) - p(ii,0) > 0){
        thj(ii) = acos((p(ii+1,0)-p(ii,0))/lj(ii));
      } else {
        thj(ii) = M_PI - acos(-(p(ii+1,0)-p(ii,0))/lj(ii));
      }
    } else {
      if (p(ii+1,0) - p(ii,0) > 0) {
        thj(ii) = -acos((p(ii+1,0)-p(ii,0))/lj(ii));
      } else {
        thj(ii) = M_PI + acos(-(p(ii+1,0) - p(ii,0))/lj(ii));
      }
    }
  }
```

```
  if (p(0,1) - p(N1-1,1) > 0){
    if (p(0,0) - p(N1-1,0) > 0){
      thj(N1-1) = acos((p(0,0)-p(N1-1,0))/lj(N1-1));
    } else {
      thj(N1-1) = M_PI - acos(-(p(0,0)-p(N1-1,0))/lj(N1-1));
    }
  } else {
    if (p(0,0)-p(N1-1,0) > 0) {
      thj(N1-1) = -acos((p(0,0)-p(N1-1,0))/lj(N1-1));
    } else {
      thj(N1-1) = M_PI + acos(-(p(0,0) - p(N1-1,0))/lj(N1-1));
    }
  }
  thj = unwrap(N1,thj);
  for (int ii = 1; ii < N1; ii++) {
    thi(ii) = 0.5*(thj(ii) + thj(ii-1));
  }
  thi(0) = 0.5*(thj(0) + thj(N1-1) + 2*M_PI);
  cout << N1 << endl;
  cout << N2 << endl;
  cout << ttf << endl;
  cout << p << endl;
}

rhoj = lj/L;
for (int ii = 0; ii<N1-1; ii++) {
  dthj(ii) = -(thi(ii+1) - thi(ii))/(rhoj(ii));
}

dthj(N1-1) = -(thi(0) - thi(N1-1) - 2*M_PI)/(rhoj(N1-1));
dthi = jtoi(N1,dthj);
```

```
// Boundary conditions
for (int ii = 0; ii < N1; ii++){
  cj(ii) = c0*(1 + tau*pow(dthj(ii)/L,1));
}
for (int ii = N1; ii < N; ii++){
  cj(ii) = 0;
}

// Boundary element method matrix creations
double dmk, ff1, ff2, ff3, f21, f22, A, B, E;
for (int mm = 0; mm < N; mm++) {
  for (int kk = 0; kk < N; kk++) {
    if (tt < 0 & mm < N1 & kk < N1) { continue;
    } else {
      A = pow(lj(kk),2);
      B = 2*lj(kk)*(n(kk,1)*(-p(kk,0)+cm(mm,0))+n(kk,0)*(p(kk,1)-cm(mm,1)));
      E = pow(cm(mm,0)-p(kk,0),2) + pow(cm(mm,1)-p(kk,1),2);
      ff3 = 4.0*A*E - B*B;
      ff1 = B/A;
      ff2 = E/A;
      ff3 = 4.0*A*E - pow(B,2);
      ff3 = sqrt(abs(ff3));
      dmk = (mm == kk) ? 0.5 : 0;
      if (ff3 < 1e-13) {
        F2(mm,kk) = -dmk;
        F1(mm,kk) = lj(kk)/(2.0*M_PI)*(log(lj(kk))+
        ...(1+ff1/2.0)*log(abs(1+ff1/2.0))-
        ...ff1/2.0*log(abs(ff1/2.0))-1);
      }
      else {
        f21 = lj(kk)*(n(kk,0)*(p(kk,0)-cm(mm,0)) +
        ...n(kk,1)*(p(kk,1)-cm(mm,1)))/(M_PI*ff3);
```

```
        f22 = atan((2*A+B)/ff3)-atan(B/ff3);
        F1(mm,kk) = lj(kk)/(4*M_PI)*(2*(log(lj(kk))-1)-ff1/2*log(abs(ff2))+
        ...(1+ff1/2)*log(abs(1+ff1+ff2)));
        F1(mm,kk) += lj(kk)/(4*M_PI)*ff3/A*f22;
        F2(mm,kk) = f21*f22 - dmk;
      }
    }
  }
}
mm = F2*cj;
dcj = F1.colPivHouseholderQr().solve(mm);

for(int ii=0; ii < N1; ii++) {
  vnj(ii) = a*a*d*dcj(ii);
}

// Evolution of the interface
vni = jtoi(N1,vnj);
for (int ii = 0; ii < N1-1; ii++) {
  dvnj(ii) = (vni(ii+1) - vni(ii))/(rhoj(ii));
  rhoi(ii+1) = 0.5*(rhoj(ii+1) + rhoj(ii));
}
rhoi(0) = rhoj(0)/2;
dvnj(N1-1) = (vni(0)-vni(N1-1))/(rhoj(N1-1));
mmj = dthj*vnj;
dL = integj(N1,rhoj,mmj);

for (int ii = 0; ii < N1; ii++) {
  ff =  integj(ii,rhoj,unif)*dL - integj(ii,rhoj,mmj);
  dtthj(ii) = -1/L*dvnj(ii) - 1/L*dthj(ii)*ff;
}
dtthi = jtoi(N1,dtthj);
```

```
  ll(tt) = L;

  L += dt*dL;

  vnit(tt) = vni(0);
  thit(tt) = thi(0);
  //p.row(0) << xx0 - integj(tt,unif*dt,vnit*sin(thit)),yy0 +
  ...integj(tt,unif*dt,vnit*cos(thit));
  p.row(0) << p(0,0) - dt*vni(0)*sin(thi(0)), p(0,1) + dt*vni(0)*cos(thi(0));
  tt += 1; tt0 += 1;
  ll(tt) = L;
  thi += dt*dtthi;
  for (int ii = 1; ii < N1; ii++) {
    p.row(ii) << p(0,0) + L*cinteg(ii+1,rhoj,thi),p(0,1)
    ...+ L*sinteg(ii+1,rhoj,thi);
  }
  // Storing data
  if (tt0%100 == 0 || tt0 == 1)
    stgdata << p << endl;
    ttstg += 1;
}

/////////////////

stgdata.close();
cout << p << endl;
cout << cm << endl;
cout << lj << endl;
cout << cj << endl;
cout << dcj << endl;
cout << vni << endl;
```

```
  cout << dthi << endl;
  cout << ll << endl;
  cout << dL << "  " << L << endl;
  ofstream myfile;
  myfile.open("data.txt");
  myfile << tt0 << endl;
  myfile << p << endl;
  myfile.close();
}
```

References

[1] W.T. Ang. *A Beginner's Course in Boundary Element Methods*. Universal Publishers, 2007. ISBN: 9781581129748. URL: https://books.google.es/books?id=c46cVdAJKZ0C.

[2] Folma Buss, J. Paul Luzio, and John Kendrick-Jones. "Myosin VI, an Actin Motor for Membrane Traffic and Cell Migration". In: *Traffic* 3.12 (2002), pp. 851–858. DOI: 10.1034/j.1600-0854.2002.31201.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1034/j.1600-0854.2002.31201.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1034/j.1600-0854.2002.31201.x.

[3] K. C. Holmes et al. "Myosin VI: cellular functions and motor properties". In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 359.1452 (2004), pp. 1931–1944. DOI: 10.1098/rstb.2004.1563. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rstb.2004.1563. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2004.1563.

[4] I. Lister et al. "Myosin VI: a multifunctional motor". In: *Biochemical Society Transactions* 32.5 (Oct. 2004), pp. 685–688. ISSN: 0300-5127. DOI: 10.1042/BST0320685. eprint: https://portlandpress.com/biochemsoctrans/article-pdf/32/5/685/537291/bst0320685.pdf. URL: https://doi.org/10.1042/BST0320685.

[5] W. W. Mullins and R. F. Sekerka. "Morphological Stability of a Particle Growing by Diffusion or Heat Flow". In: *Journal of Applied Physics* 34.2 (1963), pp. 323–329. DOI: 10.1063/1.1702607. eprint: https://doi.org/10.1063/1.1702607. URL: https://doi.org/10.1063/1.1702607.

[6] Benoit Rogez et al. "Reconstitution reveals how myosin-VI self-organises to generate a dynamic mechanism of membrane sculpting". In: *Nature Communications* 10 (July 2019). DOI: 10.1038/s41467-019-11268-9.

[7] L. M. Sander, P. Ramanlal, and E. Ben-Jacob. "Diffusion-limited aggregation as a deterministic growth process". In: *Phys. Rev. A* 32 (5 Nov. 1985), pp. 3160–3163. DOI: 10.1103/PhysRevA.32.3160. URL: https://link.aps.org/doi/10.1103/PhysRevA.32.3160.

[8] John Strain. "A boundary integral approach to unstable solidification". In: *Journal of Computational Physics* 85.2 (1989), pp. 342–389. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(89)90155-1. URL: http://www.sciencedirect.com/science/article/pii/0021999189901551.