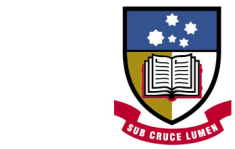


THE UNIVERSITY
*of* ADELAIDE

# An Off-lattice Computational Model for the Growth of Saccharomyces Cerevisiae

by

Isaac Nakone

In fulfilment of the requirements for the degree of

**Bachelor of Mathematical Sciences - Honours**

March 2025

Department of Mathematical Sciences

School of Computer and Mathematical Sciences

**The University of Adelaide**

# Contents

# List of Tables

# List of Figures

# Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Isaac Nakone 8/3/2025

# Acknowledgements

The \noindent command and the underlying \vspace command is personal preference here and definitely not required. Just something that I preferred after typing out my acknowledgements.

Aute eu laborum enim nulla excepteur id labore ipsum tempor sit nisi. Sunt quis commodo exercitation voluptate Lorem Lorem magna eu magna eu eu irure est minim. Ipsum voluptate consectetur in nulla tempor est eu. Labore esse in id reprehenderit. Voluptate enim id et dolor labore mollit aliqua nulla reprehenderit reprehenderit incididunt duis aliquip adipisicing. Qui ex elit laborum laboris ad. Magna nulla do amet proident ad Lorem et eiusmod occaecat quis laboris sint laborum excepteur. Laboris voluptate commodo laboris officia irure dolore deserunt amet reprehenderit mollit sint eu elit. Irure amet dolore nostrud Lorem sint labore quis ad eiusmod aliquip nulla sunt in proident. Irure reprehenderit aliqua minim velit et irure sunt dolor proident occaecat sunt sunt reprehenderit. Laborum ut dolore do culpa ullamco reprehenderit est excepteur pariatur veniam aliquip amet occaecat. Reprehenderit tempor amet nulla velit laborum.

Dolore aliqua nostrud cillum voluptate minim do reprehenderit reprehenderit do. Quis dolor deserunt esse quis tempor. Anim amet eu ipsum officia occaecat officia minim ad voluptate nisi. Cupidatat reprehenderit irure aliquip in anim aliquip ea. Deserunt sit esse laboris sunt occaecat sunt aute fugiat amet veniam cupidatat do. Ea officia tempor nostrud duis mollit ea sunt sint sint anim do aute laboris. Quis velit laboris sint Lorem eiusmod in nostrud nisi nisi cupidatat nulla. Exercitation quis aliqua ipsum ut laborum velit pariatur irure nulla sint consequat do minim adipisicing.

# Abstract

Here's what it's all about! Labore minim irure occaecat fugiat nulla sint labore et laborum eiusmod. In Lorem voluptate in do do aliqua labore velit nisi velit cupidatat deserunt. Ut aliqua officia magna id sit incididunt nulla. Mollit eu ad aute ullamco deserunt fugiat sint eiusmod aliqua cupidatat. Velit id ut est consequat duis velit. Amet quis proident culpa ea.

Cupidatat enim velit Lorem duis. Eiusmod id laboris anim nulla ad. Exercitation qui anim occaecat quis fugiat. Nisi sit minim fugiat fugiat culpa do aute consequat. Exercitation exercitation non voluptate in labore do. Consectetur sint consectetur id quis mollit.

Eu laboris deserunt dolore quis ut qui qui cupidatat irure. Est sit enim officia labore esse et duis magna incididunt nisi eiusmod officia voluptate. Consequat magna dolore laboris officia dolore. Tempor esse magna commodo ipsum aliqua aliqua commodo do cupidatat veniam velit aliquip dolore ad. Do veniam fugiat dolore aliquip esse ex nisi id cupidatat culpa nisi sit esse. Minim non est anim ad laborum velit commodo eu est. Ad labore occaecat exercitation dolor non dolor id ullamco.

Commodo non velit velit exercitation officia anim officia officia ullamco officia ad. Enim esse adipisicing in deserunt quis voluptate pariatur tempor. Ipsum magna minim ipsum consequat amet minim enim ea. Esse enim do aute commodo occaecat eiusmod quis laborum exercitation consectetur consequat mollit id. Occaecat anim sunt mollit duis non commodo non consequat eu aliquip. Occaecat eu quis minim commodo. Consequat officia velit anim reprehenderit Lorem nulla aute magna culpa magna.

Ipsum Lorem et occaecat ipsum mollit. Excepteur duis laborum amet mollit enim adipisicing ipsum minim aliqua deserunt anim sit dolor. Anim ad incididunt nisi eu.

# Introduction

- Biology of two types of yeast!
- Write chapter 2/ 3 and 4 now

The art of mathematical biology is to capture a biological phenomenon with the simplest model possible. Another concern which appears in the connection between applied and pure mathematics, is that of ensuring the model that one selects is amenable to rigorous analysis. This further reinforces the requirement that the model be as simple as possible, but no simpler. Indeed, a model that is sufficiently crystallized so as to capture the essential details of a physical effect, will necessarily not capture everything observable, but has more chance to be useful, since it directs our thinking away from extraneous features.

What is essential in the study of biological systems? One essential aspect is the phenomenon of growth. Growth, when seen on its own, without reference to biology, is simply "change over time". The standard tool for this is calculus, which in its more developed form, is called differential geometry. A cell colony will be modelled here as a subset of $\mathbb{R}^n$ where $n = 1, 2, 3$. In order for that set to "change over time", namely, for the cell to be parametrized by a real number $t \in \mathbb{R}_{\geq 0}$, the geometry of the colony needs to be represented by a finite list of parameters. The most simple way to do this, which comprises one of the models developed here, is to represent each cell as a disk in $\mathbb{R}^2$ with radius $r = r(t) \in \mathbb{R}_{\geq 0}$ centered at the position $(x(t), y(t)) \in \mathbb{R}^2$. The advantage to this approach is that it is more computationally efficient as compared to models which represent each cell as a deformable polygon, for instance.

The question remains: how can cell division be achieved within mathematical growth models? It does not seem obvious except via the continual addition of new parameters, such as new cell radii and positions. As we will see, taking seriously the question of mitosis from a topological standpoint will yield a generalized model for systems in which geometric changes and topological changes can be captured simultaneously. Why is the topology important in the case of biological growth?

As a motivating example, consider the graph of the function $f(x) = x^2$. Now consider the time-dependent set given by $C(t) = f^{-1}(\{t\})$ where $t \in \mathbb{R}_{\geq 0}$ is a time parameter. When $t = 0$, $C(0) = \{0\}$ (only one element), but when $t = 1$, $C(1) = \{-1, +1\}$, which has two elements, and, likewise for $t = 2$ where $C(t) = \{-\sqrt{2}, +\sqrt{2}\}$. This example shows that a catastrophic change (a bifurcation) can occur in $C(t)$ during a smooth change in $t$. A subtle modification to this model is to consider $f_t(x) = x^2 - t$ and fix $C(t) = f_t^{-1}(\{0\})$. That is, instead of sliding up the query point $t$, we slide down the whole smooth manifold given by $(x, f_t(x))$ such that $x \in \mathbb{R}$. The set $C(t)$ is actually called a level-0 set. We will just call this a level set of $f_t$.

We can consider level sets of functions from $\mathbb{R}^n$ to $\mathbb{R}$ for $n = 2, 3$ as well. In these cases, the level sets are, respectively, level curves and level surfaces. A analogous example for $n = 2$, is the set of functions $f_t(x, y) = x^2 + y^2 - t$, the level curves of which look like circles centered at the origin of radius $\sqrt{t}$.

This level of generalization will not be sufficient for our purposes, since the type of manifolds that we will be dealing with will not in general be representable as graphs of functions. To see this, consider the fact that a vertical line in the $xt$-plane is a perfectly reasonable representation of a stationary cell, and yet there's no function (of $x$) that has a vertical line graph. What is required is that $C(t) = \{p \in M_t \mid \text{the last component of } p \text{ is } 0\}$ where $M_t$ is a time dependent smooth manifold. The requirement that $M_t$ is smooth for all $t$ is imposed so that the theory of smooth manifolds can be brought to bear on the problem. What we will consider are smooth manifolds $M_t \subset \mathbb{R}^{n+1}$ where $n$ is the dimension in which the colony exists. For instance, a two-dimensional colony would be represented by

$$C(t) = \{p \in M_t \subset \mathbb{R}^3 \mid p_3 = 0\}. \tag{1}$$

A three-dimensional colony would be represented by

$$C(t) = \{p \in M_t \subset \mathbb{R}^4 \mid p_4 = 0\}. \tag{2}$$

In the subdiscipline of abstract algebra called ring theory, it is typical to consider the polynomial ring over the real field in two variables $\mathbb{R}[x, y]$ wherein the typical element looks like

$$p(x, y) = \sum_{n,m} a_{nm} x^n y^m, \tag{3}$$

where $n, m$ are non-negative integers, and $a_{nm}$ is a real coefficient. The set $\mathbb{R}[x, y]$ is closed under multiplication and addition because polynomial multiplication and addition yields another real polynomial. In this thesis, a cell will be modeled as a level set

of the quartic given by

$$f(x, y) = ax^4 + bx^3 + cx^2 + dx + e+ \tag{4}$$

At the beginning, we imagine a colony of yeast cells that is restricted to move along a straight line. The colony is therefore modeled as a set of real numbers $C \subset \mathbb{R}$. Like an archipelago of small islands, the colony as a whole is composed of closed sets, $C_j$ where $j \in \mathbb{N}$ indexes over the cells. Therefore, the colony is a disjoint union of these individual cells,

$$C = \coprod_{j=1}^{N} C_j$$

and $N$ is the total cell count. Closed sets (in the standard topology on $\mathbb{R}$) are chosen to represent the cells for the technical reason that a point (singleton) may also constitute a cell. In fact, we further restrict each cell to a closed interval $C_j = [a_j, b_j]$.

As we shall see in general, all we require is that each cell have no holes. Another way of saying this is that each cell is homeomorphic to a singleton (which works in $\mathbb{R}^2$ and $\mathbb{R}^3$ as well). Finally, the closed interval $[a_j, b_j]$ will be called a parametrization of the cell $C_j$: this is important to note for the generalization to $\mathbb{R}^2$ and $\mathbb{R}^3$, where parametrizations must also be constructed.

The mechanism of mitosis must account for the fact that several cells can undergo fission at the same time, or more aptly, they undergo mitosis independently. The most general formulation, which is also simple to implement computationally is the addition of a non-intersecting singleton $\{x\}$ to the set $C$. Note, that this mechanism is chosen principally for the fact that it conserves the Lebesgue measure of the colony,

$$l(C \cup \{x\}) = l(C),$$

where $C \cup \{x\}$ is the colony after mitosis has occurred, since singletons have measure 0.

Of course, some restrictions must be applied to the choice of $x \in \mathbb{R} \setminus C$. Since, we are always working in Euclidean spaces (for practical scenarios), we may as well require that $x$ is close to $C$ in the sense that for all $c \in C$ we have that $d(c, x) < \delta$ for some small positive amount $\delta$ where $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ is the Euclidean metric on $\mathbb{R}$ (which is easily generalized to higher dimensions). This parameter $\delta$ represents how the colony spreads out.

A profitable aspect of this model for cell colonies is that it unifies the topological aspects of the colony (it is always homeomorphic to a finite set of points or the empty set), and the measure theoretic properties (one can speak of getting more cells without

changing the total measure). This means we can separate the dynamics of cell division from the dynamics of the geometric growth of the cells, i.e. the change in $a_j$ and $b_j$. One simple way to do this is to supply an ordinary differential equation for the measure of the colony $V = l(C)$, and equip this with a discrete time update equation for number of cells.

To move us closer to a manageable computer implementation of cell colony growth, we consider the simple growth model given by,

$$\frac{dV(t)}{dt} = g(t),$$
$$N_{n+1} = 2N_n,$$

where $g(t)$ is a growth function. Now we need apply some equation for how each cell grows. For sake of simplicity, say that for all cells $j$, the cell measure $V_j$ grows as

$$\frac{dV_j(t)}{dt} = g_j(t), \quad t \geq m_j,$$

where $g_j(t)$ is a cellular growth function such that $V_j(t) \to V_f$ is the final cellular volume, and $m_j$ is the birth time of the cell. This actually is not enough to specify the whole colony geometry. So how do we pin down the geometry?

The geometry and how its dynamics evolve of course depends on the chosen parametrization of each cell. For our purposes, we will consider $C_j = \bar{B}(x_j, \varepsilon_j)$ where

$$\bar{B}(x_j, \varepsilon_j) = \{x \in \mathbb{R} : d(x_j, x) \leq \varepsilon_j\},$$

which is called the closed ball centered on $x_j$ of radius $\varepsilon_j$. That means $a_j = x_j - \varepsilon_j$, and $b_j = x_j + \varepsilon_j$. It is more convenient to use this parametrization since closed balls are defined in higher dimensions as well. Another important benefit to the closed ball is that whenever $\varepsilon_j = 0$, $\bar{B}(x_j, 0) = \{x_j\}$. Also, the volume of each cell is simply $V_j = 2\varepsilon_j$ which tells us that the volume is completely independent of the cell center position $x_j$.

Now we easily obtain a nice formula for $\varepsilon_j(t)$ defined for $t \geq m_j$ as

$$\varepsilon_j(t) = \frac{1}{2}\int_{m_j}^{t} g_j(s)ds.$$

But, recall, since the $C_j$ are each disjoint, $V$ is given by

$$V = l(C) = \sum_{j=1}^{N} l(C_j) = \sum_{j=1}^{N} V_j.$$

This applies to the time derivative too, yielding

$$\frac{dV(t)}{dt} = \sum_{j=1}^{N} \frac{dV_j(t)}{dt}.$$

This means that the colony and cell growth functions must be related by the following,

$$g(t) = \sum_{j=1}^{N} g_j(t).$$

Our functions $g_j(t)$ were defined for $t \in [m_j, +\infty)$ which means they have different domains. To make the analysis easier, we build these functions by taking linear combinations of basis hat functions (defined for $t \in [0, +\infty)$). The basis functions have compact support and are given piecewise by,

$$\varphi_i(t) = \begin{cases} \frac{t-t_{i-1}}{h}, & t \in [t_{i-1}, t_i] \\ \frac{t_{i+1}-t}{h}, & t \in [t_i, t_{i+1}] \\ 0, & \text{otherwise,} \end{cases}$$

where $h$ is the smallest time step, $t_0 = 0$ and $t_i = ih$ for $i \in \mathbb{N}$. Hence, each $g_j(t)$ can be extended to a definition on all of $\mathbb{R}_{\geq 0}$ by

$$\tilde{g}_j(t) = \sum_{i=1}^{\infty} g_j(t_i)\varphi_i(t) = \sum_{i=1}^{\infty} g_{ij}\varphi_i(t).$$

Now that the $g_{ij}$ are defined on the same domain, we can further restrict to $i \leq T$ where $T$ is the total number of time steps. This means that the time dependence is now fixed by a finite number of parameters, $g_{ij}$. Summing over $j$, we get that

$$\tilde{g}(t) = \sum_{j=1}^{N} \sum_{i=1}^{T} g_{ij}\varphi_i(t).$$

# Chapter 1

# Description of the model

## 1.1 An invitation to Signed Distance Fields

The use of signed distance fields (SDFs) to model organic sufaces is a time honoured graphical technique used, for example, by Pixar Animation Studios to model hair in *The Incredibles* (see **?**). The idea is to define a function which represents the closest distance from the query point to a point on the surface of the object that is to be represented. If the query point is outside, the SDF is positive, the SDF is zero on the surface and negative inside. SDFs can be rendered within traditional graphics pipelines (such as OpenGL or Vulkan) using raymarching, a method that takes place within shader programs and is therefore meshless. The formulae defining SDFs for common 2D and 3D shapes are easy to find online, see **?**. Whilst the simulations herein are done using 2D SDFs, a quick 3D primer is given below.
To motivate the primary mechanism by which cells will undergo mitosis in this thesis, we consider a toy example in which the equations for two spheres undergo a catastrophic topological change as one parameter changes. We start by considering the equations for two spheres which begin as coincident and move apart as the parameter $a$ becomes larger. In order to combine the first equation

$$f_1(x, y, z) = \sqrt{(x + a)^2 + y^2 + z^2} - r,$$

with the second equation

$$f_2(x, y, z) = \sqrt{(x - a)^2 + y^2 + z^2} - r,$$

we require a smooth combination function. We construct the combined SDF using what is called a "union" in the graphics community (see **?**). This is the pointwise minimum

$$f_{\text{union}}(x, y, z) = \min(f_1(x, y, z), f_2(x, y, z)).$$

To get smooth transition between the cells as they come apart we use smoothmin which is defined by a smoothness parameter $k$, as in

$$\text{smoothmin}(x_1, x_2) = -k \log\left(e^{-x_1/k} + e^{-x_2/k}\right). \tag{1.1}$$

As shown in Figure **??**, we have a smooth splitting of a cell as the parameter $a$ ranges from 0.0 to 5.0. Here $r$ is the nominal sphere radii, and $k$ is a smoothing parameter. The larger $k$ is, the more smoothly the two curves cling to each other. We plot the level-0 surface of the smooth union SDF using MATLAB's `isosurface` function.

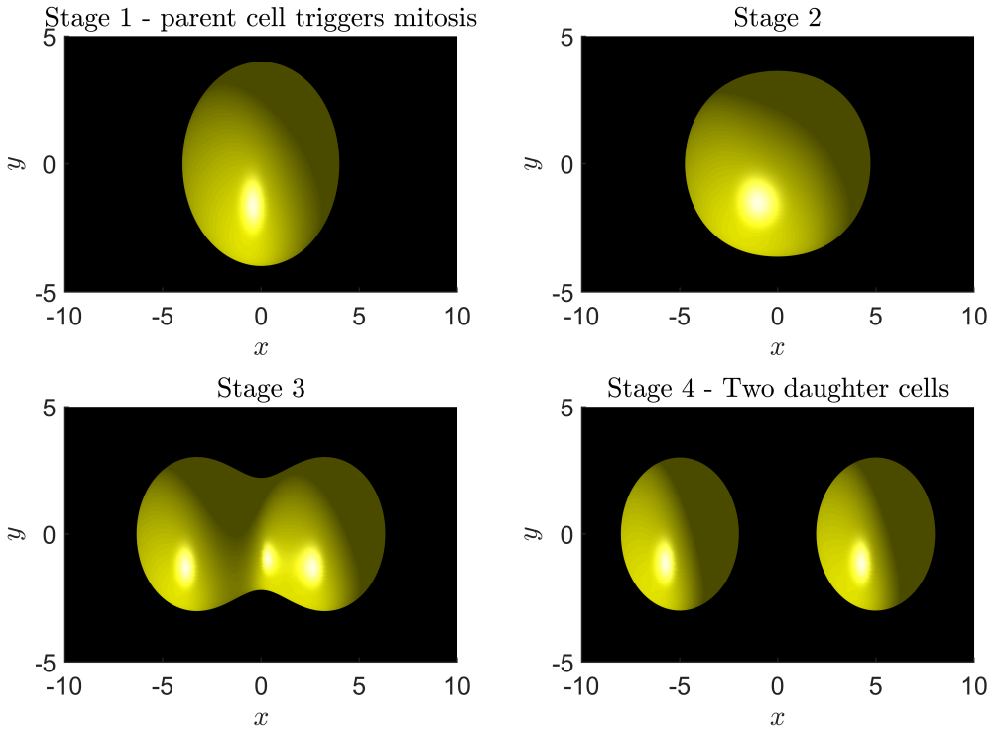


Figure 1.1: A toy example of mitosis using implicit equations for spheres in 3D. Stage 1 is $a = 0.0$, Stage 2 is $a = 1.67$, Stage 3 is $a = 3.33$, Stage 4 is $a = 5.0$

It is also possible to get the intersection of two SDFs using a smoothmax function.

## 1.2 Modeling yeast cells with ellipses

Cell colonies can be built up by combining the SDFs of the individual cells using a cumulative smoothmin. We employ the main aspect of smoothmin, which is that

$$\text{smoothmin}(f_3, \text{smoothmin}(f_1, f_2)) = -k \log\left(\sum_{j=1}^{3} e^{-f_j/k}\right), \tag{1.2}$$

therefore we can accumulate smoothmins easily using

$$\text{smoothmin}(f_1, \ldots, f_N) = -k \log\left(\sum_{j=1}^{N} e^{-f_j/k}\right), \tag{1.3}$$

The individual cells are modelled using the approximate signed distance field for an ellipse which is given at `https://www.shadertoy.com/view/4XfBDl`. I have provided the vectorized MATLAB implementation below,

```
function sdf = ellisoid_sdf(x, y, theta, r, X, Y)
    semi_minor_axis = min(r);
    semi_major_axis = max(r);
    c = sqrt(semi_major_axis^2 - semi_minor_axis^2);
    %Geometrical parameter
    center = [x,y]';
    query_point = [X(:), Y(:)]';
    R_z = rotation_matrix(theta);
    l1 = vecnorm(pagemtimes(R_z , query_point - center)
        - [c;0], 2, 1);
    l2 = vecnorm(pagemtimes(R_z , query_point - center)
        + [c;0], 2, 1);
    l = mean([l1', l2'],2);
    sdf = reshape(l', size(X)) - semi_major_axis;
end
```

Note that I multiply each local coordinate `query_point - center` by the rotation matrix `R_z` in order to rotate the cell counter-clockwise by $\theta$ radians. The rotation matrix $R_z$ is given by

$$R_z = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{1.4}$$

which is a so-called passive rotation matrix because we are rotating the coordinate system by $\theta$ CW which has the effect of rotating the ellipse by $\theta$ CCW.

where $R$ is the radius of the circle. If the filled in circle, the disk, is required, then producing a surf plot of the function in the region where $f(x, y) \geq 0$ is sufficient. This can be achieved by modifying the value of $f(x, y)$ to be `nan` wherever $f(x, y) < 0$ so that MATLAB's `surf` function does not plot it.

For approximately circular cells, each individual cell is modeled via the equation of a circle centered at position $(x_j(t), y_j(t))$ where $j$ indexes over the current total number of cells. The equation for a single cell is therefore given by

$$f_j(x, y; t) = R_j^2 - \left[(x - x_j(t))^2 + (y - y_j(t))^2\right],$$

where $R_j$ is the radius of the $j$-th cell. We will discuss how multiple cells can be blended together into one colony level implicit curve. Given the implicit curve for cell 1 to cell $N = N(t)$, respectively $f_1(x, y; t), \cdots, f_N(x, y; t)$, we blend them with the following transformation

$$\tilde{\Phi}(x, y; t) = \frac{1}{k} \ln \left[\frac{1}{N} \sum_{j=1}^{N} e^{k f_j(x,y;t)}\right],$$

where $\tilde{\Phi}(x, y; t)$ is related to the colony microscopic density, $k$ is a parameter related to the level of smoothing between the curves, $N$ is the total number of cells. If all of the $f_j(x, y; t) = 0$ then we recover that $\tilde{\Phi}(x, y; t) = 0$. Of course, we don't yet have the actual colony microscopic density, because the numerical values taken by $\tilde{\Phi}(x, y; t)$ are essentially meaningless. The microscopic density $\Phi(x, y; t)$ is obtained by translating and scaling $\tilde{\Phi}(x, y; t)$ such that $\Phi(x, y; t)$ takes values on the interval $[0, 1]$. We then interpret 1 as the maximum biomass density of the cell at each cell center.

## 1.3 Mitosis: new cells from old

The addition of new cells can be easily achieved by incorporating additional implicit curves and, hence, modifying the colony microscopic density $\Phi(x, y; t)$. In the case of circular cells which undergo effectively symmetric mitosis, it is necessary to add a cells at a slight offset so that contact forces (explained in the following section) can take effect. In order to store the data associated with each cell, a final number of cells $N_f$ is designated as a constant maximum number of cells which allows for the preallocation of the data associated with the cell centers and other state information (if necessary).

One of the desired features of the model, was to achieve cell division (which can be thought of as a catastrophic change in the colony topology) without sacrificing the smoothness of $\Phi(x, y; t)$. This has been achieved by starting the new cell at a vanishingly close distance to its parent cell, and recovering smoothness through the blending of implicit curves imposed in the construction of $\Phi(x, y; t)$. This is possibly a new idea

in the field of agent based models, where the addition of daughter cells is usually not even continuous (for example, in some off-lattice ABMs new cells simply appear beside old ones). Here, the two daughter cells move apart via contact forces between the cell centers and the smooth division follows.

## 1.4 Modeling interactions between cells

In standard particle dynamic simulations, say gravitational models, the particle positions are modeled using Newton's second law. This means the state of an $N$ particle simulation in $2D$ requires $2N$ positions and $2N$ velocity values. In overdamped, low-inertia regime, which is often used in cell off-lattice ABMs, it is sufficient to ignore changes in the velocity and only construct equations for the change in position over time.

## 1.5 Tuning the model parameters

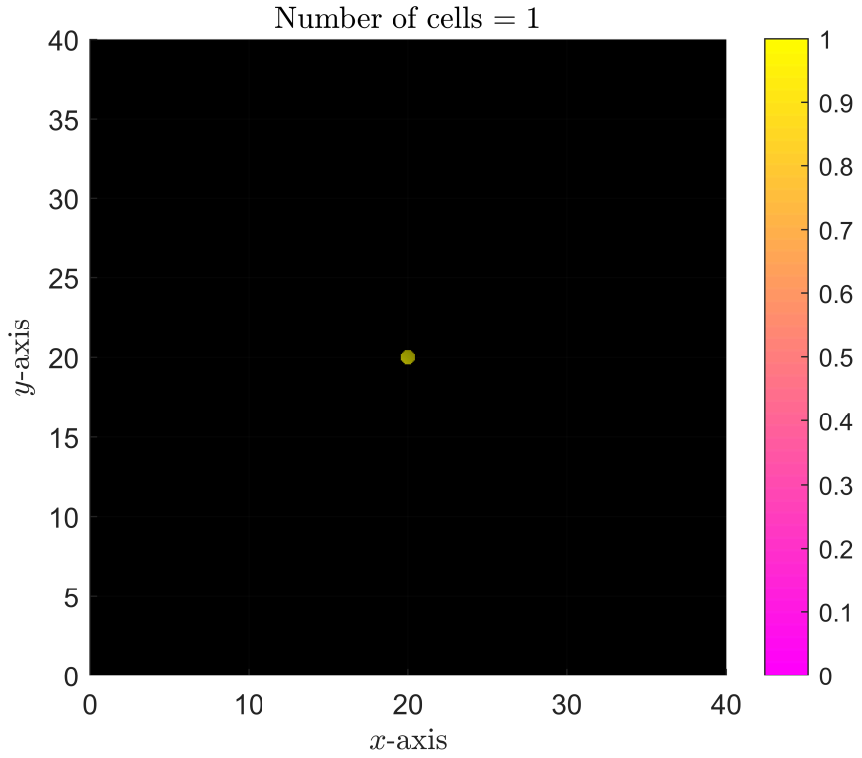


Figure 1.2: A starting configuration consisting of a cell with radius 1 unit equals $3.5\mu m$

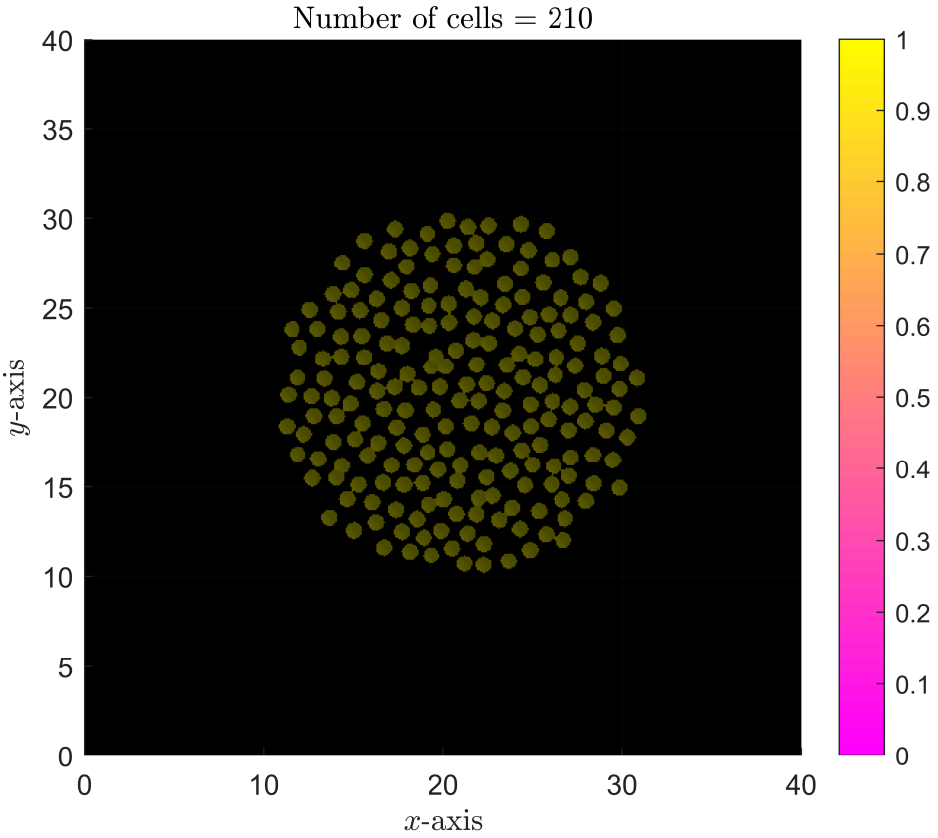


Figure 1.3: The same colony at 210 cells

## 1.6 Adding in a nutrient field

## 1.7 Simulating $N > 1000$ cells with a nutrient field

Explain how I have used hash mapping to deal with cell collisions.

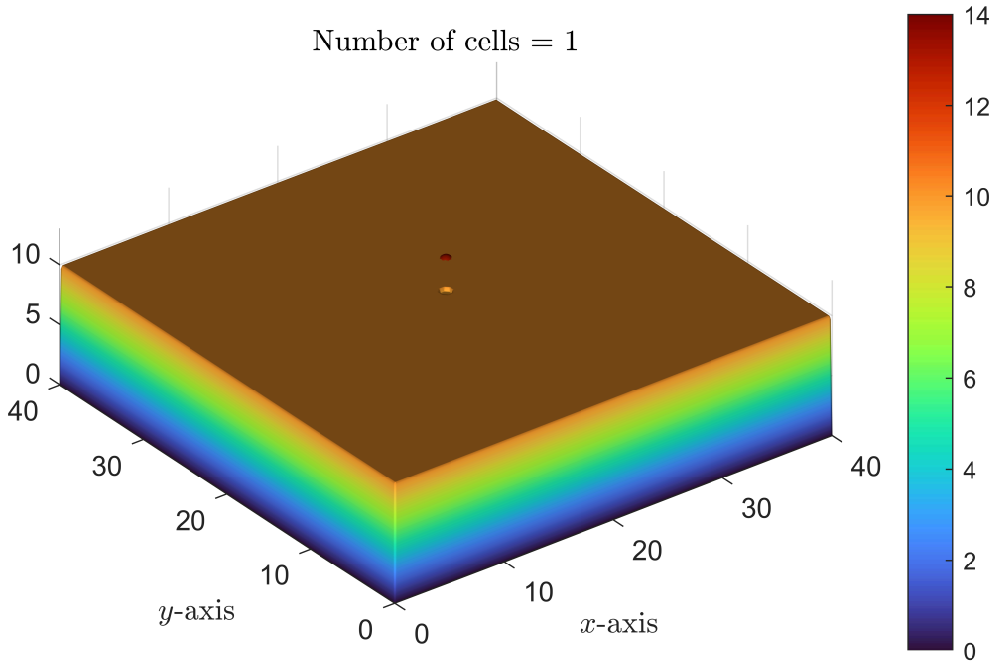


Figure 1.4: A starting configuration consisting of a cell with radius 1 unit equals $3.5\mu m$

## 1.8 Calculating the compactness metric

A fully grown colony will in general not be perfectly circular in shape. In order to measure the roundness of the colony we use the metric used for roundness in image processing

$$\Psi = \frac{P^2}{4\pi A}, \tag{1.5}$$

where $\Psi \in [0, 1]$ is 1 for a perfect circle and can get to 0 for highly non-round shapes, $A$ is the colony area, and $P$ is the colony perimeter. When the cells are undergoing mitosis, we are left with the issue of calculating roundness of the blended "pill" shape geometry of two cells right before splitting. In affect, it will be necessary to calculate the length of

## 1.9 Collision Detection

Referring to the supplied figures, we see that a force is **F** in Figure **??** can be a representation of an applied force from another cell acting on the current cell. We set

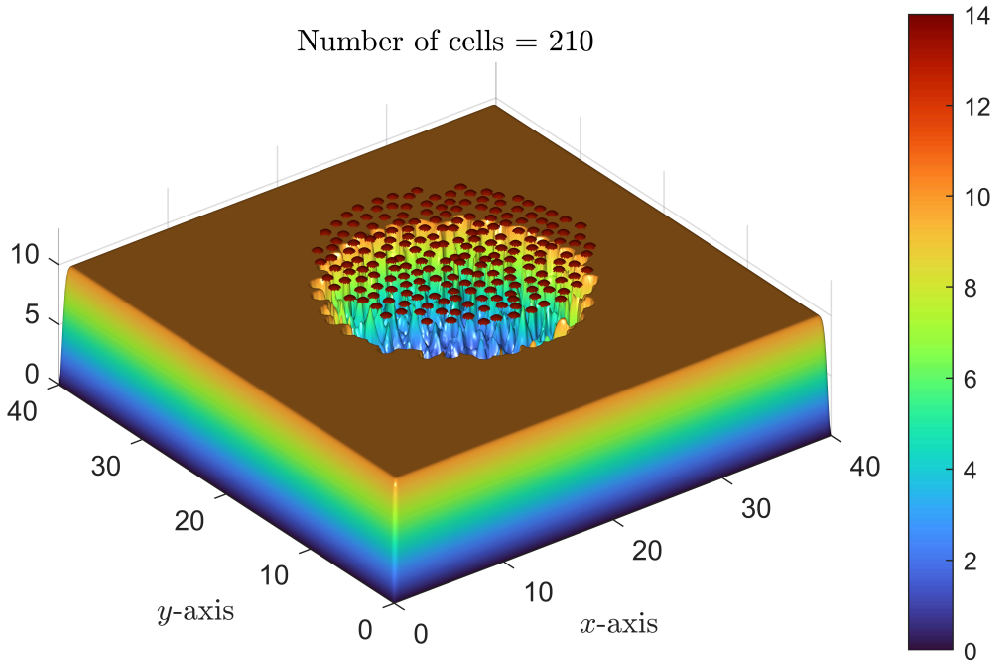


Figure 1.5: The same colony at 210 cells

out to find forces on the two point mass that will produce equivalent force balance in the $(t, n)$ coordinate system.

We replace the force $\mathbf{F}$ with $\mathbf{F}_1$ and $\mathbf{F}_2$ acting on mass 1 and 2, respectively. From force balance

$$\mathbf{F}_1 + \mathbf{F}_2 = \mathbf{F},$$

and, from moment balance with anti-clockwise positive,

$$l_F F_n = d(F_{2,n} - F_{1,n}).$$

In our case, the impinging force will be from another cell, so we start by making the assumption that there is no tangential force, i.e. frictionless slipping. With this we can derive simultaneous equations for $F_{1,n}$ and $F_{2,n}$ which have the solution

$$F_{1,n} = \left(\frac{d - l_F}{2d}\right) F_n,$$

$$F_{2,n} = \left(\frac{d + l_F}{2d}\right) F_n.$$

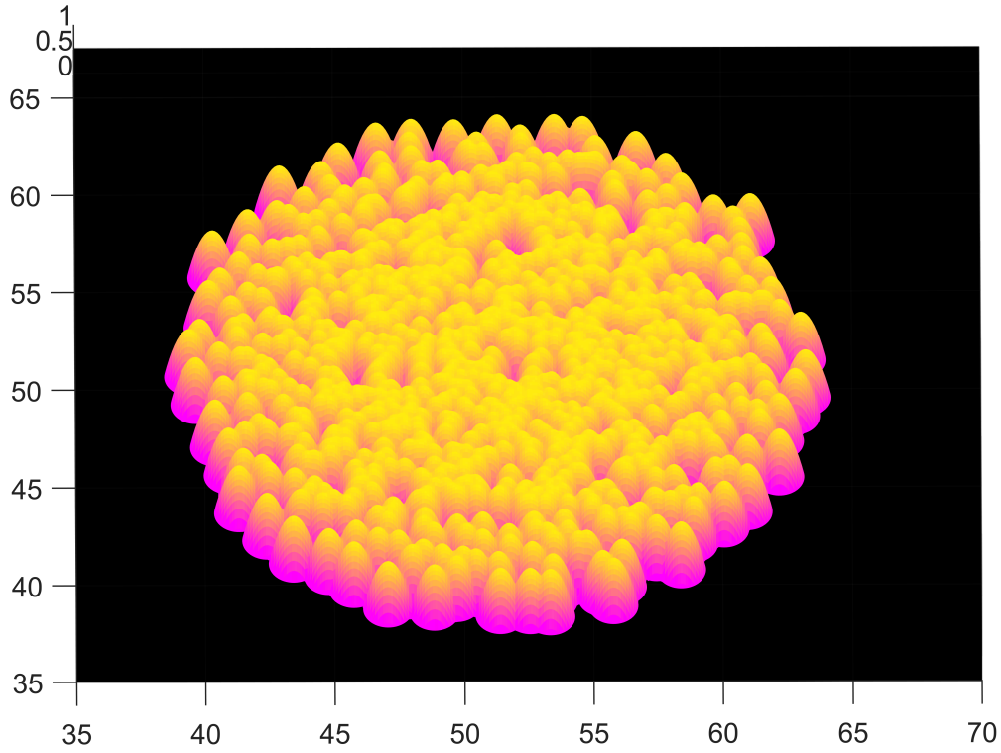


Figure 1.6: A cell colony with 1065 cells

Assuming, once again, that $F_{1,t} = F_{2,t} = F_t = 0$ we can obtain the $(x, y)$ components by doing a coordinate transformation by the angle $-\theta_C = -\arctan(y_2 - y_1, x_2 - x_1)$ where $(x_1, y_1)$ and $(x_2, y_2)$ are the mass positions, respectively. So, we have that the components of the force are given by

$$\begin{bmatrix} F_{1,x} \\ F_{1,y} \end{bmatrix} = \begin{bmatrix} \cos(-\theta_C) & -\sin(-\theta_C) \\ \sin(-\theta_C) & \cos(-\theta_C) \end{bmatrix} \begin{bmatrix} -F_{1,n} \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta_C & \sin\theta_C \\ -\sin\theta_C & \cos\theta_C \end{bmatrix} \begin{bmatrix} -\left(\frac{d-l_F}{2d}\right) F_n \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} F_{2,x} \\ F_{2,y} \end{bmatrix} = \begin{bmatrix} \cos(-\theta_C) & -\sin(-\theta_C) \\ \sin(-\theta_C) & \cos(-\theta_C) \end{bmatrix} \begin{bmatrix} F_{2,n} \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta_C & \sin\theta_C \\ -\sin\theta_C & \cos\theta_C \end{bmatrix} \begin{bmatrix} \left(\frac{d+l_F}{2d}\right) F_n \\ 0 \end{bmatrix}.$$

We solve this as

$$\mathbf{F}_1 = -\left(\frac{d-l_F}{2d}\right) F_n \cos\theta_C \hat{\mathbf{i}} + \left(\frac{d-l_F}{2d}\right) F_n \sin\theta_C \hat{\mathbf{j}},$$

$$\mathbf{F}_2 = \left(\frac{d+l_F}{2d}\right) F_n \cos\theta_C \hat{\mathbf{i}} - \left(\frac{d+l_F}{2d}\right) F_n \sin\theta_C \hat{\mathbf{j}},$$

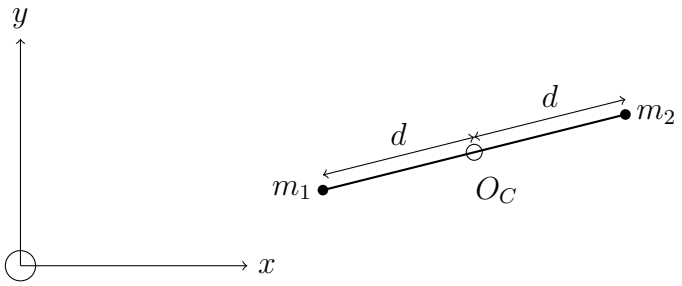


Figure 1.7: Shows a rod shaped backbone of an elliptical cell made from two masses $m_1 = m_2 = m$ connected by a massless rod of length $2d$

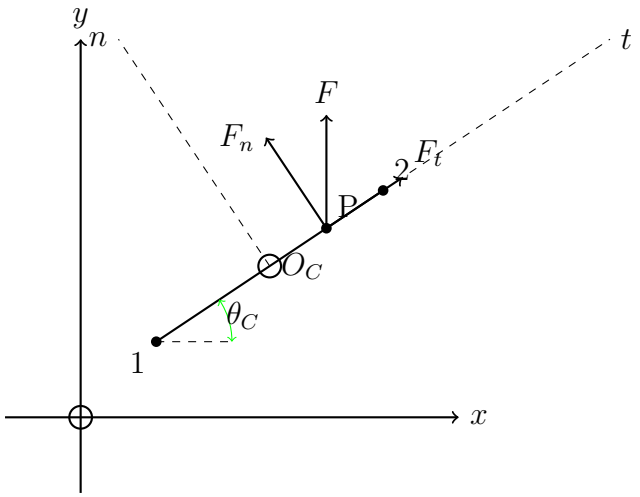


Figure 1.8: Shows a rod shaped cell which is being impinged upon by a contact force **F** resolved into a tangential $F_t$ and normal component $F_n$.

## 1.10 Approximate Bayesian Computation (ABC) for the inverse problem

The traditional statement of Bayes' Theorem goes like

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{1.6}$$

where $A$ and $B$ are outcomes and the terms are given by

- $P(A|B)$ is the posterior probability; the probability of $A$ given $B$,
- $P(B|A)$ is the conditional probability
- $P(A)$ is the prior probability; probability of observing event $A$
- $P(B)$ is the marginal probability; probability of observing event $B$

# Chapter 2

# Applying the model

# Chapter 3

# Extending the model

## 3.1 Cell-cell collisions with constrained dynamics

Earlier, it was mentioned that the intersection of cells could be found by taking a smoothmax. To find the overlapping area the sum of the grid points in the intersection can be taken and then multiplied by the grid square area $h_x h_y$. Whilst it is more or less trivial to calculate the overlapping area, ensuring that this area remains zero throughout the simulation is more involved. We employ the technique explained by **?** based on constrained dynamics. The constraint in this case is that the area $C$ remains 0 for all times,

$$C(\mathbf{q}) = 0, \tag{3.1}$$

where $\mathbf{q}$ is the concatenated state vector of the system. One should be aware that, if we have multiple colonies, it will be required to have pairwise constraints forcing no overlap between each of the constitiuent colonies. For the purpose of simplicity our state vector $\mathbf{q}$ will just take into account position and orientation and not size of cells. Later on, this will be generalized. Therefore $\mathbf{q}$ is produced by concatenating over $\mathbf{q}_j$ into one $3N \times 1$ column vector, where

$$\mathbf{q}_j = \begin{bmatrix} x_j \\ y_j \\ \theta_j \end{bmatrix}. \tag{3.2}$$

Of course, the number of variables per cell can be larger but this comes at a cost in computational time. Recall that the intersecting area is secretly a function of the cell state coordinates $\mathbf{q}_j$ since we are taking a smoothmax of the SDFs of each cell. In other words,

$$f_{\text{intersect}}(x, y, \mathbf{q}) = \text{smoothmax}(f_1(x, y, \mathbf{q}), \ldots, f_N(x, y, \mathbf{q})),$$

for concreteness, we write out the formula for smoothmax which is the negative smoothmin of the negatives of the SDFs or

$$f_{\text{intersect}}(x, y, \mathbf{q}) = -\text{smoothmin}(-f_1(x, y, \mathbf{q}), \ldots, -f_N(x, y, \mathbf{q})).$$

In full this boils down to,

$$f_{\text{intersect}}(x, y, \mathbf{q}) = k \log\left(\sum_{j=1}^{N} e^{f_j(x,y,\mathbf{q})/k}\right).$$

Now we assume the state is given by five variables for full generality:

$$\mathbf{q}_j(t) = \begin{bmatrix} x_j(t) \\ y_j(t) \\ \theta_j(t) \\ b_j(t) \\ R_j(t) \end{bmatrix}, \tag{3.3}$$

where $b(t)$ is the semi-major axis length and $R_j(t) \in (0, 1]$ is the aspect ratio of the elliptical cell so the semi-minor axis is given by $a_j(t) = R_j(t)b_j(t)$. Why are we going to the effort to write out $C(\mathbf{q})$ in full? Because we need to compute the Jacobian of $C(\mathbf{q})$ with respect to $\mathbf{q}$ in order to carry out the constrained dynamics algorithm. The upshot is that smoothmax is differentiable whereas maximum is not differentiable.

We define the region in $\mathbb{R}^2$ to integrate over by

$$\Omega(\mathbf{q}) = \{(x, y) \in \mathbb{R}^2 \mid f_{\text{intersect}}(x, y, \mathbf{q}) \leq 0\}, \tag{3.4}$$

so that the area of the overlapping region is simply a two-dimensional integral over $\Omega(\mathbf{q})$ of 1,

$$C(\mathbf{q}) = \int\int_{\Omega(\mathbf{q})} dxdy. \tag{3.5}$$

Now we break up the integral into a sum over simply connected components (SCC) so that we can safely apply Leibniz's rule

$$C(\mathbf{q}) = \sum_{i \in SCC(\mathbf{q})} \int\int_{\Omega_i(\mathbf{q})} dxdy. \tag{3.6}$$

Leibniz's rule tells us how to carry out a total derivative of $\int\int_{\Omega_i(\mathbf{q})} dxdy$ with respect to $t$ which determines $\mathbf{q}$. Let's take that total derivative now to attain

$$\frac{d}{dt}\int\int_{\Omega_i(\mathbf{q})} dxdy = \int\int_{\Omega_i(\mathbf{q})} \frac{\partial(1)}{\partial t} dxdy + \int_{\partial\Omega_i(\mathbf{q})} (1)\mathbf{v}_{\partial\Omega_i(\mathbf{q})} \cdot \hat{\mathbf{n}} dl, \tag{3.7}$$

where $\mathbf{v}_{\partial\Omega_i(\mathbf{q})}$ is the "Eulerian" velocity of the boundary of the $i$-th SCC. All of this can be avoided if we integrate over a smoothstep which is defined in our region.

$$C(\mathbf{q}) = \int\int_D \left[\frac{1}{2} + \frac{1}{2}\tanh\left(-\frac{1}{K} f_{\text{intersect}}(x, y, \mathbf{q})\right)\right] dxdy, \tag{3.8}$$

where $D$ is the entire domain of the pertri dish.

$$\frac{\partial f_{\text{intersect}}(x, y, \mathbf{q})}{\partial \mathbf{q}} = \frac{\sum_{j=1}^{N} \frac{\partial f_j(x,y,\mathbf{q})}{\partial \mathbf{q}} e^{f_j(x,y,\mathbf{q})/k}}{\sum_{j=1}^{N} e^{f_j(x,y,\mathbf{q})/k}}.$$

Conveniently, computing the Jacobian, has turned into $N$ problems that are easier to solve individually. Namely computing $\frac{\partial f_j(\mathbf{q})}{\partial \mathbf{q}}$. This requires us to express the SDF for an ellipse in terms of the query point $(x, y)$, the center of the cell $(x_j, y_j)$, and its orientation $\theta_j$. This is given in terms of $l_j^-(\mathbf{q}, x, y)$ and $l_j^+(\mathbf{q}, x, y)$ which are given in the ellipse formula. We use MATLAB's symbolic toolbox to compute the ellipse Jacobian $\frac{\partial f_j(x,y,\mathbf{q})}{\partial \mathbf{q}}$ and return a function that can take numerical inputs using `matlabFunction(J,"File","ellipseJacobian")`. With that we can compute Jacobian of the constraint using

$$\frac{\partial C(\mathbf{q})}{\partial \mathbf{q}} = -\frac{1}{2K}\int\int_D \left[1 - \tanh^2\left(-\frac{f_{\text{intersect}}(x, y, \mathbf{q})}{K}\right)\right] \frac{\partial f_{\text{intersect}}(x, y, \mathbf{q})}{\partial \mathbf{q}} dxdy. \tag{3.9}$$

Now that all the derivatives have been computed analytically, we can carry out the computation of the integral over $x$ and $y$ using a numerical integral in MATLAB. From now we use $f$ instead of $f_{\text{intersect}}$ for brevity, and we pass to index notation, instead expression the $\beta$-th component of $J$ as

$$J_\beta = -\frac{1}{2K}\int\int_D \left[1 - \tanh^2\left(-\frac{f(x, y, \mathbf{q})}{K}\right)\right] \frac{\partial f(x, y, \mathbf{q})}{\partial q_\beta} dxdy. \tag{3.10}$$

For the computation of constraint forces, we need to further compute the total derivative of the Jacobian with respect to time. This turns out to be realted to the Hessian matrix of $C(\mathbf{q})$ (for no explicit time dependence), as

$$\dot{J}_\beta = \sum_{\alpha=1}^{5N} \dot{q}_\alpha \frac{\partial^2 C}{\partial q_\alpha \partial q_\beta} = \sum_{\alpha=1}^{5N} \dot{q}_\alpha H_{\alpha,\beta},$$

We need to compute how the Hessian of $C$ can be written in terms of the Hessian and Jacobian of $f$.

$$\frac{\partial^2 C}{\partial q_\alpha \partial q_\beta} = \frac{\partial}{\partial q_\alpha} J_\beta \tag{3.11}$$

Crunching the calculuations we get

$$\frac{\partial}{\partial q_\alpha} J_\beta = -\frac{1}{4K^2} \int\int_D \left[1 - \tanh^2\left(-\frac{f}{K}\right)\right] \left[\tanh\left(-\frac{f}{K}\right) \frac{\partial f}{\partial q_\alpha}\frac{\partial f}{\partial q_\beta} + K\frac{\partial^2 f}{\partial q_\alpha \partial q_\beta}\right] dxdy \tag{3.12}$$

Luckily, the Hessian is built into MATLAB's symbolic toolbox, so we can just call `matlabFunction(H,"File","ellipseHessian")` and the function to compute the Hessian is saved into our working directory. Note that the function `ellipseHessian` computes a $5 \times 5 \times N_{\text{grid}} \times N_{\text{grid}}$ matrix. We call it for each cell $j \in \{1, \ldots, N\}$ but recall that for a given value of $\mathbf{q}$, the Hessian still has $(x, y)$ as free field variables. In order to integrate the field data, we use the cell-wise Jacobian field $J^j_\beta(x, y)$ and the cell-wise Hessian field $H^j_{\alpha,\beta}(x, y)$ to calculate the overall Hessian field for $f$

$$\frac{\partial^2 f}{\partial q_\alpha \partial q_\beta} = \left(\frac{1}{k}\right) \left[\frac{\left(\sum_{j=1}^N E_j\right)\left(\sum_{j=1}^N E_j(kH^j_{\alpha,\beta} + J^j_\alpha J^j_\beta)\right) - \left(\sum_{j=1}^N E_j J^j_\alpha\right)\left(\sum_{j=1}^N E_j J^j_\beta\right)}{\left(\sum_{j=1}^N E_j\right)^2}\right],$$

where $E_j = e^{f_j/k}$ is used for short. To actually compute the overall integrals for $\frac{\partial C}{\partial q_\beta}$ and $\frac{\partial^2 C}{\partial q_\alpha \partial q_\beta}$ we use MATLAB's `trapz` function the following way

```
trapz(y_lin,trapz(x_lin,integrandJacobian,3),2);
```

or, in the case of the Hessian,

```
trapz(y_lin,trapz(x_lin,integrandHessian,4), 3);
```

where the integrand in the Jacobian cas is given by

$$F_\beta(x, y) = \left(\frac{T^2 - 1}{2K}\right) \frac{\sum_{j=1}^N J^j_\beta E_j}{\sum_{j=1}^N E_j},$$

where $T = \tanh\left(-\frac{f(x,y,\mathbf{q})}{K}\right)$ and, in the case of the Hessian

$$G_{\alpha,\beta}(x, y) = \left(\frac{T^2 - 1}{4K^2}\right)\left(T\left(\frac{\sum_{j=1}^N J^j_\alpha E_j}{\sum_{j=1}^N E_j}\right)\left(\frac{\sum_{j=1}^N J^j_\beta E_j}{\sum_{j=1}^N E_j}\right) + K\frac{\partial^2 f}{\partial q_\alpha \partial q_\beta}\right),$$

which, after some serious simplification becomes

$$G_{\alpha,\beta}(x, y) = \frac{T^2 - 1}{4Kk\left(\sum_{j=1}^N E_j\right)^2} \sum_{n=1}^N \sum_{m=1}^N E_n E_m \left[\left(\frac{kT - K}{K}\right) J^n_\alpha J^m_\beta + J^m_\alpha J^m_\beta + kH^m_{\alpha,\beta}\right]$$

To make things neat we replace $B = \frac{T^2-1}{4K^2}$, $\hat{E} = \sum_{j=1}^{N} E_j$ and we subsitute the ratio of the smoothstep and smoothmax parameters $S = K/k$ to attain,

$$G_{\alpha,\beta}(x,y) = \frac{B}{\hat{E}^2}\sum_{n=1}^{N}\sum_{m=1}^{N} E_n E_m \Big[ S(kH^m_{\alpha,\beta} + J^m_\alpha J^m_\beta) + (T-S)J^n_\alpha J^m_\beta \Big],$$

Thus we can write

$$J_\beta = \int\int_D F_\beta(x,y)dxdy,$$

$$\dot{J}_\beta = \sum_{\alpha=1}^{5N} \dot{q}_\alpha \int\int_D G_{\alpha,\beta}(x,y)dxdy,$$

Now we subsitute to obtain the final integral formulae:

$$J_\beta = \int\int_D \frac{B}{\hat{E}S}\sum_{j=1}^{N} J^j_\beta E_j dxdy,$$

$$\dot{J}_\beta = \int\int_D \frac{B}{\hat{E}^2}\sum_{\alpha=1}^{5N}\sum_{n=1}^{N}\sum_{m=1}^{N} \dot{q}_\alpha E_n E_m \Big[ S(kH^m_{\alpha,\beta} + J^m_\alpha J^m_\beta) + (T-S)J^n_\alpha J^m_\beta \Big] dxdy.$$

We set out to calculate the following scalar $A = JWJ^t$ where $W = M^{-1}$ is the inverse mass matrix of our dynmical system. Note, that since we have only one constraint our goal is to solve for one Lagrange multiplier $\lambda$ which is given as

$$A\lambda = -\sum_{\beta=1}^{5N} \dot{J}_\beta \dot{q}_\beta - JWQ - \kappa_s C - \kappa_d \dot{C},$$

Recall $C = \frac{1}{2}\int\int_D (1+T)dxdy$ and $\dot{C} = \sum_{\beta=1}^{5N} \dot{q}_\beta J_\beta$. This means we can write out our equation for $\lambda$ explicitely

$$\left(\sum_{\alpha=1}^{5N}\sum_{\beta=1}^{5N} J_\alpha W_{\alpha,\beta} J_\beta\right)\lambda = -\sum_{\beta=1}^{5N} \dot{J}_\beta \dot{q}_\beta - \sum_{\alpha=1}^{5N}\sum_{\beta=1}^{5N} J_\alpha W_{\alpha,\beta} Q_\beta - \kappa_s \frac{1}{2}\int\int_D (1+T)dxdy - \kappa_d \sum_{\beta=1}^{5N} \dot{q}_\beta J_\beta,$$

Note that its convenient from a computatational point of view to bring the quadriple sum into the integral and then evaluate this using `trapz`:

$$\int\int_D \frac{B}{\hat{E}^2}\sum_{\alpha=1}^{5N}\sum_{\beta=1}^{5N}\sum_{n=1}^{N}\sum_{m=1}^{N} (\dot{q}_\alpha \dot{q}_\beta E_n E_m) \Big[ S(kH^m_{\alpha,\beta} + J^m_\alpha J^m_\beta) + (T-S)J^n_\alpha J^m_\beta \Big] dxdy.$$

where the term $\dot{q}_\alpha \dot{q}_\beta E_n E_m$ is a $(5N) \times (5N) \times N \times N \times N_{\text{grid}} \times N_{\text{grid}}$ array. Now, of course the integral we are after is the following

$$I^{l,m,n}_{\alpha,\beta} = \int\int_D \frac{B_l}{E_l^2}(E_n E_m)\Big[S(kH^m_{\alpha,\beta} + J^m_\alpha J^m_\beta) + (T_l - S)J^n_\alpha J^m_\beta\Big]dxdy.$$

Observing this formula, we can note that $\hat{E}$ has been replaced by $E_l$. This is actually an improvement since $\hat{E}$ was only ever part of the smoothmax approximation, we replace it by an arbitrary $E_l$ which must be picked prior to evaluating the integral based on the maximum. I have subscripted $B_l$ and $T_l$ similarly as they depend on $E_l$. The main reason for this was due to difficulty in evaluating the integral over a reciprical sum when $N$ was arbitrary. Note that if we evaluate this integral symbolically in preprocessing we can essentially divide the amount of computational work by $10^6$ for a $1000 \times 1000$ grid. Even now, the compute time scales as $N^4$ for $N$ cells. This is still intractable. A significant optimisation can be made when we realise that, from the point of view of a partial derivative, a cell's SDF is not affected by the coordinates of another cell. Another way to put this is that the Jacobian and Hessian for a given cell $j$ depend only on the attributes of that cell and all the other partial derivatives will vanish. This actually reduces the ammount to compute by a factor of $N^2$ because we only need to sum over the number of attributes per cell ($N_{\text{attrib}} = 5$ in our case) squared. A modern laptop can perform floating point operations in the order of tens of GFLOPS. Supposing the calculation of $I^{l,m,n}_{\alpha,\beta}$ requires 1000 floating point operations per $(m, n)$ pair. If we try to push to $N = 1000$ yeast cells, then we will be looking at a second compute time per time step assuming the PC used operates at 1 GFLOP. This is fine for small $N$ but the problem doesn't scale well. In any case, with the current optimisations, we have a tractable simulaton.

It is worth noting that further optimisations can be made if we employ a spatial hash map or a AABB filter which avoids computing matrix elements between cells which are not even nearby each other. For example, if cells 3 and 22 are further than $d$ apart where $d$ is the pair's maximum cell diameter then we can set the matrix element $I^{l,3,22}_{\alpha,\beta} = I^{l,22,3}_{\alpha,\beta} = 0$ straight away (Note sure about this). We carry the full computation in the test phase and add the filter optimisation later.

# Conclusions

Reprehenderit voluptate fugiat qui mollit nulla occaecat officia consectetur non voluptate culpa cillum tempor aliqua. Laborum tempor mollit excepteur laboris. Laboris sint et in exercitation nostrud laboris ipsum eiusmod. Fugiat incididunt eiusmod incididunt Lorem. Ipsum cillum Lorem aliqua ex occaecat occaecat ex elit veniam tempor. Reprehenderit duis esse enim aute officia. Ut culpa reprehenderit proident nostrud qui anim duis cillum eu incididunt. Voluptate aliqua in aliquip pariatur amet do consectetur. Incididunt proident sunt eu officia do ex qui ea mollit. Esse et excepteur incididunt enim excepteur do ullamco sit proident officia aliquip. Mollit eu ipsum ex quis est esse id deserunt est. Amet occaecat tempor dolor sit nulla. Veniam quis elit Lorem aliqua consectetur consequat occaecat.

Ea cillum mollit ut tempor consectetur ea consectetur. Magna quis in consequat labore do labore officia incididunt Lorem eu. Minim tempor officia consequat veniam. Ex anim et officia voluptate enim aliquip ad ad veniam mollit fugiat tempor est. Sint pariatur consequat Lorem incididunt duis reprehenderit esse exercitation cillum est id eu cupidatat. Exercitation voluptate cillum cupidatat veniam adipisicing incididunt sit et adipisicing magna deserunt tempor. Cupidatat ipsum dolore quis nisi. Culpa duis in nisi quis ex enim aliquip fugiat cillum. Labore magna dolore esse in non aliquip exercitation labore aliquip eu velit dolor quis. Nisi sit in quis quis labore.

Commodo ea esse non commodo pariatur duis ut culpa laborum. Reprehenderit excepteur consequat aliqua eu nulla esse in ex voluptate nulla. Tempor incididunt laborum anim dolor deserunt occaecat. Consequat pariatur nostrud do id voluptate incididunt sunt sint sunt ad. Esse Lorem minim in et fugiat dolor. Nostrud ad est cupidatat cillum. Officia ut occaecat fugiat adipisicing do. Id cupidatat in est laborum eiusmod minim. Dolore qui magna dolor irure proident proident eiusmod ex laboris enim aliqua. Velit laboris et quis velit ea ullamco duis cillum velit eiusmod irure non eu sit. Incididunt enim aute labore pariatur sint. Cupidatat sunt consequat voluptate fugiat consectetur minim esse. Fugiat elit eiusmod laborum consectetur velit anim officia proident.

Et excepteur sint amet occaecat qui dolor. Tempor deserunt dolore id magna consequat laborum sunt. Qui culpa est id laborum eu deserunt dolor incididunt non eiusmod et est ex. Id anim commodo nostrud tempor fugiat cillum ex culpa eu dolore in laborum

veniam. Excepteur adipisicing exercitation deserunt nulla deserunt amet officia tempor incididunt cillum qui. Fugiat quis ut incididunt enim labore dolor. In ut sunt deserunt ullamco nostrud irure eiusmod adipisicing sunt magna consequat. Fugiat sunt commodo nostrud veniam dolore id. Tempor ut Lorem enim ullamco mollit magna. Labore Lorem ipsum minim incididunt tempor velit ea mollit exercitation aliquip pariatur ea. Ipsum voluptate quis adipisicing ut non dolor.

Reprehenderit ipsum tempor labore duis tempor culpa duis non anim amet id reprehenderit aliquip. Dolore et in non aliqua. Qui ut est magna sit sit in proident et aliqua elit sint voluptate eiusmod esse. Cillum anim ea eu ullamco culpa dolore eiusmod incididunt esse.

# Appendix A

# My appendices from chapter 1

## A.1 First appendix section

Minim sint duis sunt adipisicing ad. Occaecat labore in eu ad dolor adipisicing consequat. Do occaecat Lorem consectetur quis aliquip dolore consequat aliquip proident voluptate est ipsum cillum. Dolor ea veniam adipisicing veniam do ut laborum consequat laboris reprehenderit aliquip. Excepteur Lorem dolore amet esse dolor duis adipisicing quis dolore laboris cillum voluptate. In mollit aliquip duis do aute.

## A.2 Second appendix section if required

Fugiat amet sit cillum duis. Eiusmod nisi nulla cillum laborum labore non. Officia incididunt ea elit tempor excepteur mollit eiusmod Lorem ullamco. Do cupidatat sunt nisi pariatur proident enim adipisicing pariatur officia incididunt sunt nostrud reprehenderit in. Est amet excepteur ad laborum cillum laboris Lorem officia id. Voluptate velit aute Lorem ea eiusmod veniam.

Ut laboris aute tempor duis ad irure minim ea pariatur eu est tempor mollit anim. Nulla voluptate voluptate nostrud minim exercitation ex qui excepteur sint amet in veniam. Est minim aliquip consequat aliquip aute ullamco enim et aliquip. Aliquip sunt do aute reprehenderit ullamco occaecat consequat non. Ipsum nostrud in culpa adipisicing consectetur eu ad dolore in sint est id.