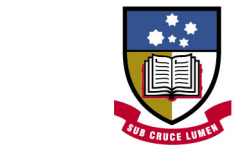

THE UNIVERSITY
of ADELAIDE

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

May 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 8/5/2025

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 δ 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 δ 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 ε_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 Petrovic et al. (2005)). 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 Quilez (2025). 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 FUSEK). 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.

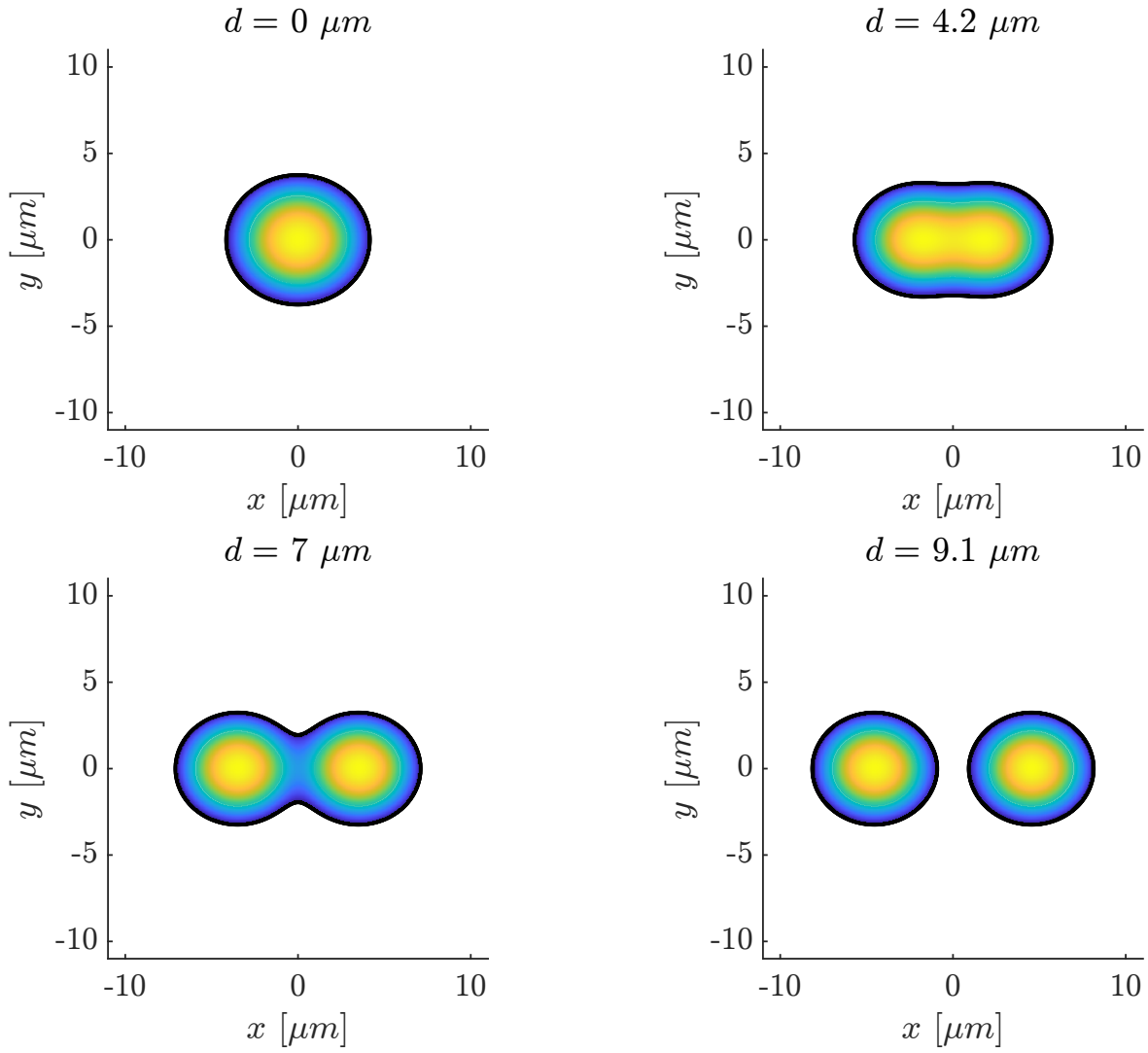

Figure 1.1: The separation d between two ellipse shapes with $s = 0.5$ and an aspect ratio of 0.9

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

1.2 Modeling yeast cells with ellipses

A signed distance field for the ellipse is used to model Baker's yeast cells. An ellipse centered at the origin with semi-major dimension a (the x intercept) and semi-minor

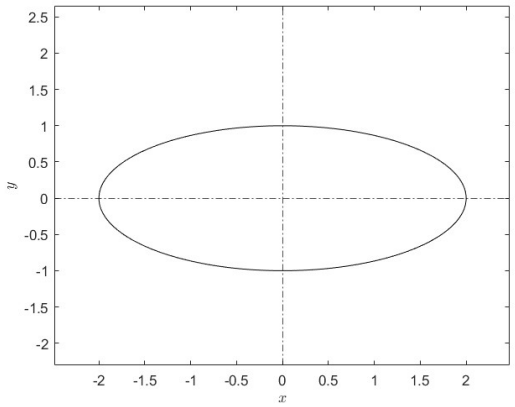

Figure 1.2: An ellipse centered at the origin with $a = 2$ and $b = 1$.

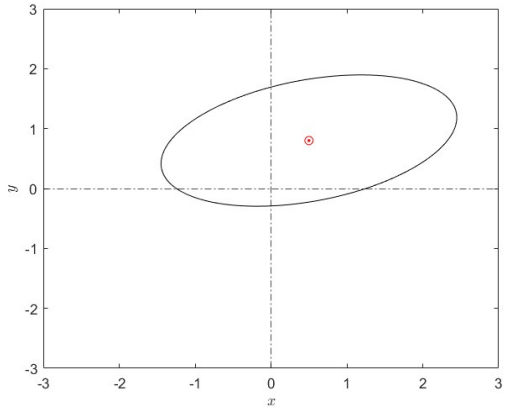

Figure 1.3: An ellipse centered at $(0.5, 0.8)$ with $a = 2$, $b = 1$ and $\theta = 15°$.

dimension b (the y intercept) has an SDF given by

$$f(x, y) = \sqrt{\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2} - 1.$$

This is not really a *distance* field because it is dimensionless but it will still be called an SDF since it produces the elliptical shape all the same. Here is our ellipse. We can also translate and rotate the ellipse, using

$$\Delta\mathbf{x}' = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \Delta\mathbf{x},$$

where $\Delta\mathbf{x} = (x - x_0)\hat{\mathbf{i}} + (y - y_0)\hat{\mathbf{j}}$ and (x_0, y_0) is the center of the ellipse. We call the components of $\Delta\mathbf{x} = \Delta x\hat{\mathbf{i}} + \Delta y\hat{\mathbf{j}}$ and develop the following formula

$$f(x,y) = \sqrt{\left[\frac{(x-x_0)\cos\theta + (y-y_0)\sin\theta}{a}\right]^2 + \left[\frac{-(x-x_0)\sin\theta + (y-y_0)\cos\theta}{b}\right]^2} - 1.$$

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

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

1.3 Cell colony as dynamical system

Each cell indexed $j \in \{1, \ldots N\}$ has five pieces of data which are enough to define globally the SDF of the cell, namely, the center coordinates (x_j, y_j), the angle of orientation θ_j, and the semi axes dimensions a_j, b_j. Each piece of data will be a function of time t. For the purpose of simplicity, we model each cell as two point masses $m_1 = m_2 = m$ connected by a spring with stiffness K. The point masses are located at $\mathbf{r}_j^{(1)}$ and $\mathbf{r}_j^{(2)}$ inside the ellipse along the major axis and symmetrically about the ellipse center. This means that the center is given by

$$\mathbf{x}_j = \frac{1}{2}\left(\mathbf{r}_j^{(1)} + \mathbf{r}_j^{(2)}\right).$$

Fixing the semi-minor axis b_j, we give the semi-major axis a_j by

$$a_j = a_0||\mathbf{r}_j^{(1)} - \mathbf{r}_j^{(2)}||.$$

We extract the orientation angle using a two argument inverse tangent function,

$$\theta_j = \arctan\left(y_j^{(2)} - y_j^{(1)}, x_j^{(2)} - x_j^{(1)}\right)$$

When the number of cells is fixed, the colony dynamics is modelled using first order EOMs with two primary forces: intracellular spring force and and intercellular

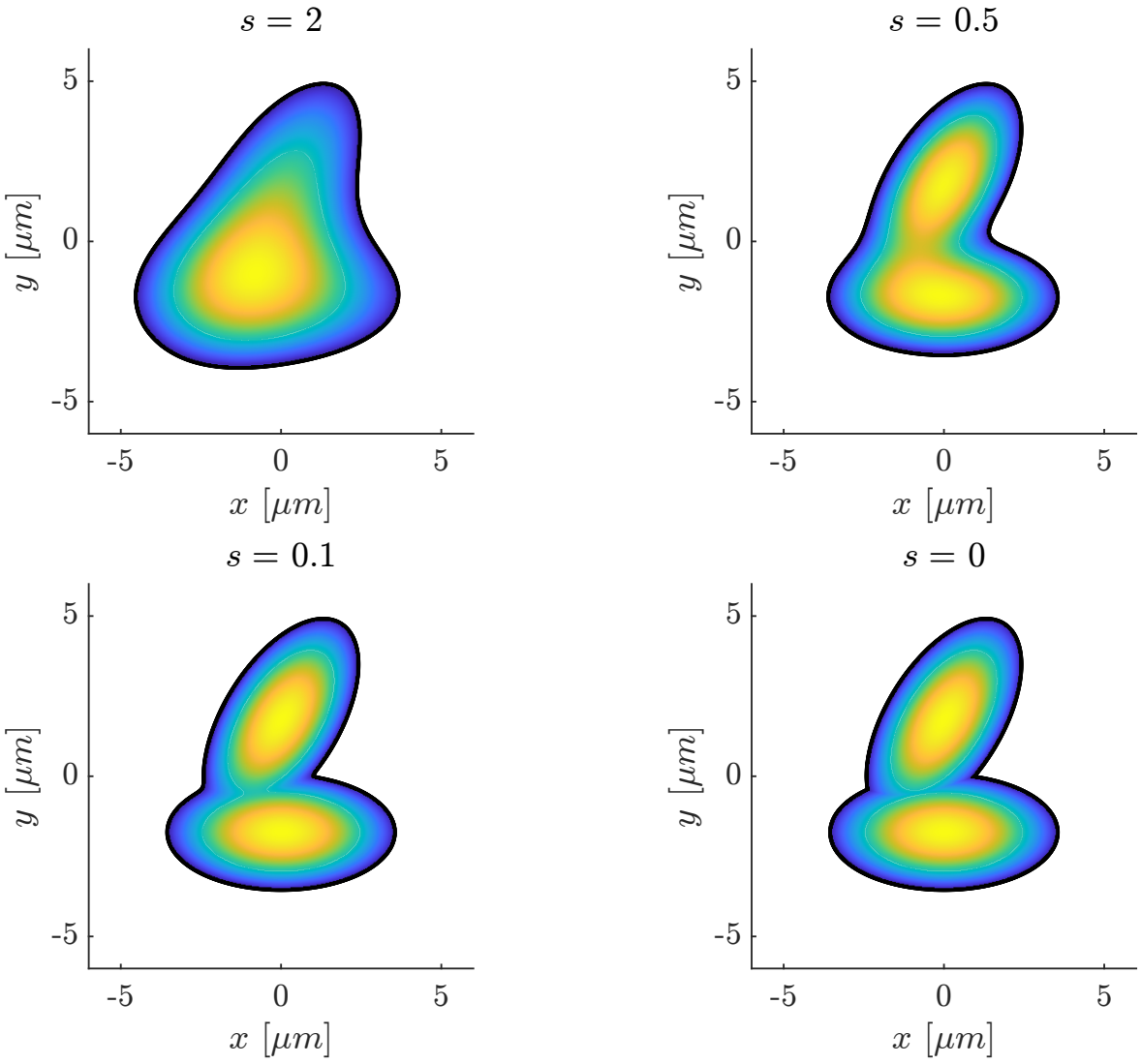

Figure 1.4: Two ellipses blended with various values of smoothness s. The bottom right figure is computed using standard minimum instead of smoothmin. Indeed, as $s \to 0$, the plot looks less smoothed across.

contact force to void overlap. We take the assumption that many authors make (include reference here) which is that inertia is negligible due to drag effects (more on this). That is the velocity is directly proportional to the force

$$\mathbf{v}_j^{(i)} = \frac{1}{\eta}\mathbf{F}_j^{(i)},$$

where $i \in \{1, 2\}$ and $j \in \{1, \ldots, N\}$ where η is an expression for the drag and $\mathbf{F}_j^{(i)}$ is the sum of the forces acting on the i-th particle of the j-th cell. Overall, the simulation

will be begun with V vertices where V is a positive power of 2. We map from local indices (i, j) to a global index n using

$$n = 2(j-1) + i,$$

which is called "row major order". We then flatten the list of x-coordinates and y-coordinates into one state vector $\mathbf{X}(t)$ given by

$$\mathbf{X}(t) = [x_1(t), \ldots, x_V(t), y_1(t), \ldots, y_V(t)]^T,$$

where (x_n, y_n) is the coordinate of the n-th vertex for $n \in \{1, \ldots, V\}$. Note that the change in the number of cells is simulated by removing constraints between the vertices. At the beginning of the simulation $x_1 = \cdots = x_V$ and $y_1 = \cdots = y_V$. The first order ordinary differential equation is phrased in terms of a mass matrix and a force function,

$$M(t, \mathbf{X})\frac{d\mathbf{X}(t)}{dt} = f(t, \mathbf{X})$$

We start by considering the spring force acting on the n-th particle due to the m-th particle given n and m are connected by springs. This is given by $\mathbf{F}_{nm}^{\text{spring}}$ as

$$\mathbf{F}_{nm}^{\text{spring}} = K(||\mathbf{x}_m - \mathbf{x}_n|| - L_{nm})\frac{\mathbf{x}_m - \mathbf{x}_n}{||\mathbf{x}_m - \mathbf{x}_n||},$$

where K is a spring constant meant to represent cell elasticity, and L_{nm} is the nominal length of the spring connecting them. There are three situations regarding edge between vertex n and m: either they are connected by a spring with $L_{nm} > 0$, they are collocated by an equality constraint or they are disconnected. If the two masses are collocated by an equality constraint, then the spring force is undefined so we must omit this. We must also omit the contact force because this has no sense for collocated vertices. The contact force between two disconnected vertices is given as

$$\mathbf{F}_{nm}^{\text{contact}} = \begin{cases} C\frac{\mathbf{x}_m - \mathbf{x}_n}{||\mathbf{x}_m - \mathbf{x}_n||}, & \text{if } ||\mathbf{x}_m - \mathbf{x}_n|| \leq d \\ \mathbf{0}, & \text{if } ||\mathbf{x}_m - \mathbf{x}_n|| > d, \end{cases}$$

which is used to ensure that vertices do not overlap past a threshold distance d. In terms of the connectivity, we can encode the fact that two vertices are connected (by a spring) in an adjacency matrix A_{nm} which is equal to 1 if they are connected and 0 otherwise. We also introduce a second matrix B_{nm} which represents when two vertices are disconnected, i.e., $B_{nm} = 1 - A_{nm}$. A third matrix is introduced for collocation $C_{nm} = 1$ if $n \neq m$ and n and m are collocated and 0 otherwise. This matrix (in fact $\tilde{C}_{nm} =\sim C_{nm}$) will be used as a logical mask to filter out `nan` values from the force matrices.

We can think of the forces (whether elastic or contact) as pairs of matrices $(F_x)_{nm}^{\text{spring}}$ and $(F_y)_{nm}^{\text{spring}}$ and similarly for the contact forces. We construct the x-component of the overall force vector

$$(f_x)_n = \sum_{m=1}^{V}((F_x)^{\text{spring}}(A\&\tilde{C}))_{nm} + \sum_{m=1}^{V}((F_x)^{\text{contact}}(B\&\tilde{C}))_{nm},$$

where $A\&\tilde{C}$ are mask matrices that ensure both A and not C are satisfied. Similarly for the y-component,

$$(f_y)_n = \sum_{m=1}^{V}((F_y)^{\text{spring}}(A\&\tilde{C}))_{nm} + \sum_{m=1}^{V}((F_y)^{\text{contact}}(B\&\tilde{C}))_{nm}.$$

The question remains: how are the mask matrices A, B and C made to change over time? Here we will do a small illustrative example with $N = 4$ cells and $V = 2N = 8$ vertices.

Initially all the vertices will be collocated at the same position (x_0, y_0). This means that the force vector should come out to zero because we essentially have one particle not interacting with anything. Let us check that $A\&\tilde{C}$ and $B\&\tilde{C}$ both vanish. C is given directly as a matrix of all ones, which says that each vertex is constrained to every other vertex. Thus $\tilde{C} = O$ where O is a 8×8 zero matrix. Initially, say $A = O$ (the zero matrix) because there are no springs. Note that this immediately says that B is a matrix of all ones. In any case, both $A\&\tilde{C} = B\&\tilde{C} = O$.

At some point, the effectively single particle starts growing away from its initial position. This is achieved by parcelling half of the vertices into one set of collocated points, and the other half into another set of collocated points. Programmatically, we select half of the points and translate them to a random nearby position to (x_0, y_0) and modify the C matrix to remove the constraints between the first and second set,

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

At this point, we should also set the nominal length of the springs as L_0 and the corresponding matrix of lengths $L_{mn} = L_0$ (the constant matrix with value L_0). As it

turns out, the action of the spring pressing the vertices apart will model the geometric growth of each elliptical cell. The A matrix will be given by

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now let's suppose that the vertices in one set undergo another mitosis event, splitting half of the particles off into a third set. The new C matrix, called C' must reflect this change. We call $C(\alpha_1, \ldots, \alpha_M) = \text{diag}(\mathbf{1}_{V/2^{\alpha_1}}, \ldots, \mathbf{1}_{V/2^{\alpha_M}})$ where $\mathbf{1}_{V/2^{\alpha_q}}$ is an all 1's matrix of dimension $V/2^{\alpha_q} \times V/2^{\alpha_q}$ where q indexes over the powers of two and represents the accumulation of division events. During a mitosis event in which the α_q-th vertex set splits, $C(\alpha_1, \alpha_2, \ldots, \alpha_M) \to C(\alpha_1, \ldots, \alpha_q + 1, \alpha_q + 1, \ldots \alpha_M)$. In other words, the α_q-th matrix splits into two matrices of half the size. In our 8×8 case, in which we divide the second vertex set, this results in the following

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The new matrix A is got by adding a connections between the left over smaller matrices, which is best understood by visualising the matrix as

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Taking another division event on the second block, we get

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

1.4 Mitosis: new cells from old

In summary, the mechanism of mitosis is a slight of hand, in the sense that each of the nodes (two nodes per cell) are already there at the simulation outset. The appearance of growth of the colony coincides with the nodes' progressive dislodgement from their equality constraints with one another. Of course, there is no reason why the model could not be modified to add nodes that were not there to begin with, though such a model would yield equivalent results.

To interpret the model mathematically, we recall that it was desireable to have a time parameter t over which the state of the colony could vary. The main goal of the thesis was to demonstrate that a system, in this case a yeast colony, which appears to acquire more dynamical variables (new coordinates for new cells for example) could be represented by a model with a fixed number of variables which are initially coincident. Philosophically, however, the model has some limitations. In reality, we know the growth of fungal species such as Baker's yeast, is strongly dependent on factors such as temperature and the environmental conditions that the colony is subjected to. This means that the dynamics of cell growth (the collision and interaction as well as the morphology) in any large colony system cannot be intrinsically defined. To put it succinctly, the dynamics of the colony is strongly coupled to the local environmental conditions.

Does this mean that we can only model biological systems completely if all the conditions are incorporated? As per the discussion in the introduction about mathematical modelling, we aim to consider the simplest viable model. This, in the case of the current work, is a model that has two components: a cell colony as discussed in the preceeding section, and a nutrient field. Indeed, their coupling determines the dynamics.
In the discussion, I will suggest ways to relax these limiting assumptions that are outside the scope of the current work but are nonetheless interesting to consider.
This model, in which the colony appears as the "unfolding" of a network of nodes follows on somewhat naturally from the theory of L-systems, in which the underlying discrete structure of a system is allowed to evolve deterministically based on rules analogous to cellular automata. A criticism which has been leveled (find relevant paper that I read a while ago) at L-systems and other deterministic models of plants for example, is what I call their intrinsic nature. That is, they effectively rely on the assumption that a biological system's rules for developement are somehow contained within it's structure (for instance, DNA). It is more fashionable nowadays, and indeed more scientifically accurate, to think of a biological system as part of a complex network of other organisms and environmental conditions which determine its morphology.
Consider a minimalistic example of modelling the motion of a spherical ball rolling around on a table. Two spatial paramters x and y, together as a tuple are enough to say where it is. However, if the ball unexpectately falls from the table, then you would suddenly require another parameter to describe the state of the system. The value of this classical example is to demonstrate for N particles and M constraints that may suddenly change, we can derive rich and unexpected dynamics.
The best we can hope for is that our parametrisation is somehow dense in the space of all possible cell colony configurations that we see experimentally. There are even some exotic cases of cellular sytems in which a single cell can have nontrivial topologies (ask Ed about the slime cells which optimise their topology based on nutrient). Therefore, future work should focus on representations of dynamic cell geometries that are as free of assumptions as is conceivable. Once an abstract enough representation of a biological system and its environment becomes available in the theory, one can use it to study and predict novel morphologies that appear in nature. Hopefully it is clear that the present work is just a suggestion of the numerous possibilities in the area of biolgical morphology viewed in the framework of "growing geometry".

1.5 Adding in a nutrient field

The nutrient field is some form of glucose mixture (agar) that is assumed to be given by a reaction-diffusion partial differential equation. That is, the nutrient concentration

$u(x, y, t)$ is given by

$$\frac{\partial u(x,y,t)}{\partial t} = D\left(\frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2}\right) - ru(x,y,t)f(x,y,t) \tag{1.4}$$

where $f(x, y, t)$ is the microscopic biomass density of the cell colony, D is a diffusion coefficient and r is a constant measuring the rate at which the cells consume nutrient. The PDE is subjected to a Dirichlet condition, i.e that the nutrient density vanishes at the boundary. In order to simulate the nutrient field numerically we use a finite difference on the square grid covering the domain. This is implemented using a 5-point stencil for space and a forward Euler method for time.

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = D\left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{h^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{h^2}\right) - ru_{i,j}^n f_{i,j}^n, \tag{1.5}$$

where i, j take on values in the interior grid points, n is a positive integer time index, h is the spatial step (same in x and y directions) and Δt is the time increment. See the figure below for an initial condition set to a sine wave product which vanishes on the boundaries.

1.6 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 compactness metric used for roundness in image processing (quote Kai use of this metric)

$$C = \frac{P^2}{4\pi A}, \tag{1.6}$$

where $C \in [1, \infty)$ is 1 for a circle and can get to large numbers for highly branching shapes, A is the colony area, and P is the colony perimeter. Both of these are calculated from the formula for the microscopic cell density which is always given by when $f(x, y, t)$ changes sign. A black and white image is produced at each time step using Matlab's function `bwboundaries` as per Kai's suggestion. The area then is given by summing up the grid squares that are inside the implicit shape givcen by f and multiplying by h^2. The perimeter is got by using `bwboundaries`, which outputs an array of points on the boundary from which the Euclidean distance between neighbouring points is found and then summed over.

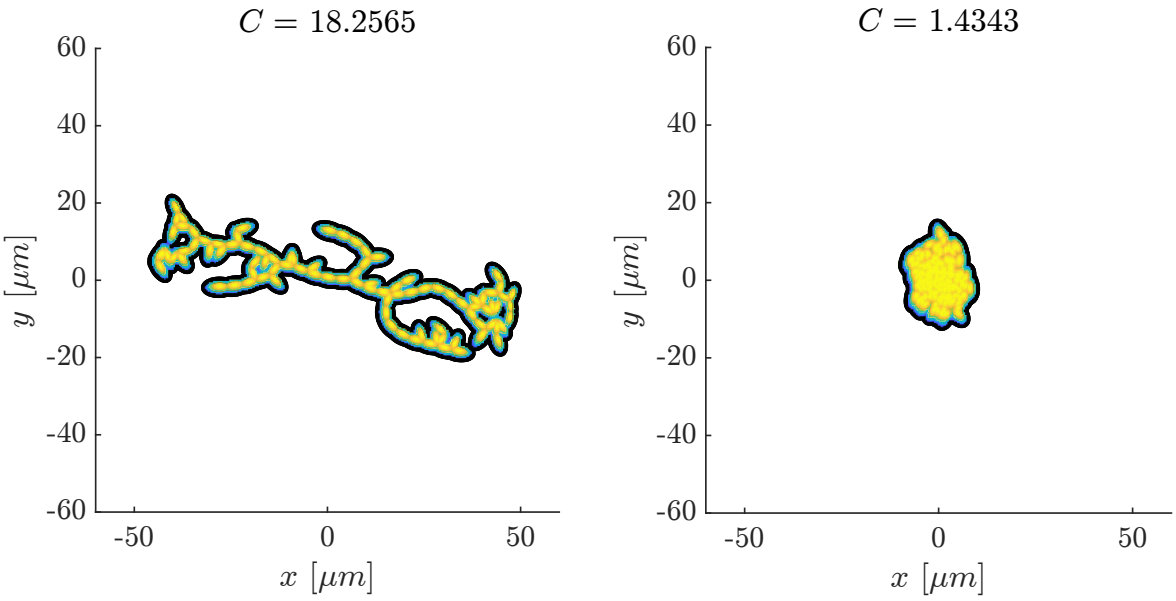

Figure 1.5: A comparsion of high and low compactness

Chapter 2

Applying the model

2.1 Non-dimensionalising the equations of motion

In order to non-dimensionalise the equations of motion (EOMs) we introduce dimensionless parameters $\mathbf{x}_i = L\hat{\mathbf{x}}_i$, $t = T\hat{t}$, $b = B\hat{b}$ and $c = C\hat{c}$, where L, T, B and C are general undetermined scalings for the independent and dependent variables, respectively. At the outset, we fix $L = L_0$ the nominal major cell diameter, and $T = \frac{1}{\mu^*}$ the reciprocal of specific growth rate for yeast, that is the $\mu^* = \mu(0)$ that appears in the formula for the total cell count (talk about ideal conditions $\mu^* = 0.4$ hour^{-1})

$$N(t) = N_0 e^{t\mu(t)}.$$

Substituting in these parameters, we obtain

$$\frac{C}{(1/\mu^*)}\frac{\partial \hat{c}}{\partial \hat{t}} = \left(\frac{DC}{L_0^2}\right)\left(\frac{\partial^2 \hat{c}}{\partial \hat{x}^2} + \frac{\partial^2 \hat{c}}{\partial \hat{y}^2}\right) - (rCB)\,\hat{c}\hat{b},$$

which results in

$$\frac{\partial \hat{c}}{\partial \hat{t}} = \left(\frac{D}{\mu^* L_0^2}\right)\left(\frac{\partial^2 \hat{c}}{\partial \hat{x}^2} + \frac{\partial^2 \hat{c}}{\partial \hat{y}^2}\right) - \left(\frac{rB}{\mu^*}\right)\hat{c}\hat{b}.$$

Call the dimensionless diffusion constant $\lambda_1 = \frac{D}{\mu^* L_0^2}$ and set $\frac{rB}{\mu^*} = 1$ which are free to do since B is unset to begin with. This means that the scaling for the biomass density is $B = \frac{\mu^*}{r}$. The reaction diffusion equation reduces to

$$\frac{\partial \hat{c}}{\partial \hat{t}} = \lambda_1\left(\frac{\partial^2 \hat{c}}{\partial \hat{x}^2} + \frac{\partial^2 \hat{c}}{\partial \hat{y}^2}\right) - \hat{c}\hat{b}.$$

Now, for the EOM for the nodes, we have

$$\begin{aligned}\frac{d\mathbf{x}_i}{dt} &= \frac{K}{\eta}\sum_{j=1}^{N}\left[-A_{ij}\left(||\mathbf{x}_i-\mathbf{x}_j||-L_0\right)\frac{\mathbf{x}_i-\mathbf{x}_j}{||\mathbf{x}_i-\mathbf{x}_j||}\right]\\ &+\frac{F}{\eta}\sum_{j=1}^{N}\left[H(R-||\mathbf{x}_i-\mathbf{x}_j||)\frac{\mathbf{x}_i-\mathbf{x}_j}{||\mathbf{x}_i-\mathbf{x}_j||}\right]\\ &+\frac{\gamma}{\eta}\nabla c(\mathbf{x}_i,t).\end{aligned}$$

Substituting the dimensionless parameters, we acquire

$$\begin{aligned}\frac{L_0}{(1/\mu^*)}\frac{d\hat{\mathbf{x}}_i}{d\hat{t}} &= \frac{KL_0}{\eta}\sum_{j=1,j\neq i}^{N}\left[-A_{ij}\left(||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||-1\right)\frac{\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j}{||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||}\right]\\ &+\frac{F}{\eta}\sum_{j=1,j\neq i}^{N}\left[H(\hat{R}-||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||)\frac{\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j}{||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||}\right]\\ &+\frac{\gamma C}{\eta L_0}\hat{\nabla}\hat{c}(\hat{\mathbf{x}}_i,\hat{t}),\end{aligned}$$

which, after rearrging, becomes,

$$\begin{aligned}\frac{d\hat{\mathbf{x}}_i}{d\hat{t}} &= \frac{K}{\eta\mu^*}\sum_{j=1,j\neq i}^{N}\left[-A_{ij}\left(||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||-1\right)\frac{\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j}{||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||}\right]\\ &+\frac{F}{\eta\mu^*L_0}\sum_{j=1,j\neq i}^{N}\left[H(\hat{R}-||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||)\frac{\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j}{||\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j||}\right]\\ &+\frac{\gamma C}{\eta\mu^*L_0^2}\hat{\nabla}\hat{c}(\hat{\mathbf{x}}_i,\hat{t}).\end{aligned}$$

Only one of the coefficients could be set to unity, but instead we choose $C = c_0 = 1$ to be the initial concentration. That leaves us with four additional parameters,

$$\lambda_2 = \frac{K}{\eta\mu^*},$$

$$\lambda_3 = \frac{F}{\eta\mu^*L_0},$$

$$\lambda_4 = \frac{R}{L_0},$$

$$\lambda_5 = \frac{\gamma c_0}{\eta\mu^*L_0^2}.$$

We also think of the biomass field as being directly equal (in units) to the total signed distance field (rather than just proportial to). So, we are left with the following EOM for the biomass field,

$$B\hat{b} = \begin{cases} -L_0\hat{g}(\hat{x},\hat{y},\hat{t}), & \text{if } \hat{g}(\hat{x},\hat{y},\hat{t}) \leq 0, \\ 0, & \text{otherwise.} \end{cases}$$

$$\hat{b} = \begin{cases} -\frac{rL_0}{\mu^*}\hat{g}(\hat{x},\hat{y},\hat{t}), & \text{if } \hat{g}(\hat{x},\hat{y},\hat{t}) \leq 0, \\ 0, & \text{otherwise.} \end{cases}$$

which leaves us with an additional parameter, $\lambda_6 = \frac{rL_0}{\mu^*}$. The final model parameter is the cell aspect ratio, λ_7.

Symbol	Expression	Descriptive name
λ_1	$\frac{D}{\mu^* L_0^2}$	diffusivity
λ_2	$\frac{K}{\eta\mu^*}$	elasticity
λ_3	$\frac{F}{\eta\mu^* L_0}$	respulsivity
λ_4	$\frac{R}{L_0}$	respulsion radius
λ_5	$\frac{\gamma c_0}{\eta\mu^* L_0^2}$	mobility
λ_6	$\frac{rL_0}{\mu^*}$	metabolic rate
λ_7	$\frac{\text{cell minor axis}}{\text{cell major axis}}$	cell aspect ratio

Table 2.1: A summary of the dimensionless model parameters

Dropping the hats on variable symbols, we arive at the final system equations of motion.

$$\boxed{\frac{\partial c}{\partial t} = \lambda_1 \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) - bc, \text{ for } (x, y) \in \Omega \setminus \partial\Omega,} \tag{2.1}$$

$$\boxed{c(x, y, t) = 1, \text{ for } (x, y) \in \partial\Omega,} \tag{2.2}$$

$$\boxed{c(x, y, 0) = 1,} \tag{2.3}$$

$$\boxed{\begin{aligned} \frac{d\mathbf{x}_i}{dt} = {} & \lambda_2 \sum_{j=1}^{N_{\text{nodes}}} \left[A_{ij} \left(1 - ||\mathbf{x}_i - \mathbf{x}_j||\right) \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||} \right] \\ & + \lambda_3 \sum_{j=1}^{N_{\text{nodes}}} \left[H(\lambda_4 - ||\mathbf{x}_i - \mathbf{x}_j||) \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||} \right] \\ & + \lambda_5 \nabla c(\mathbf{x}_i, t), \text{ for } i \in \{1, ..., N_{\text{nodes}}\}, \end{aligned}} \tag{2.4}$$

$$\boxed{\mathbf{x}_1(0) = (-0.5, 0), \ \mathbf{x}_2(0) = (0.5, 0),} \tag{2.5}$$

$$\boxed{N_{\text{nodes}} = N_0 e^{t\mu(t)}, \text{where } N_0 = 1,} \tag{2.6}$$

$$\boxed{\mu(t) = \frac{1}{N_{\text{nodes}}} \sum_{j=1}^{N_{\text{nodes}}} c(\mathbf{x}_j, t),} \tag{2.7}$$

$$\boxed{b(x, y, t) = \begin{cases} -\lambda_6 g(x, y, t), & \text{if } g(x, y, t) \leq 0, \\ 0, & \text{otherwise,} \end{cases}} \tag{2.8}$$

$$\boxed{g(x, y, t) = \min_{k \in \{1, ..., N_{\text{cells}}\}} f_k(x, y, t),} \tag{2.9}$$

$$\boxed{\begin{aligned} f_k(x, y, t) = {} & \left[\frac{(x - x_{c,k}(t)) \cos\theta_k + (y - y_{c,k}(t)) \sin\theta_k}{a_k(t)} \right]^2 \\ & + \left[\frac{-(x - x_{c,k}(t)) \sin\theta_k + (y - y_{c,k}(t)) \cos\theta_k}{b_k(t)} \right]^2 - 1, \end{aligned}} \tag{2.10}$$

$$\boxed{a_k(t) = ||\mathbf{x}_{j_1} - \mathbf{x}_{j_2}||, \text{ where cell } k \text{ is comprised of nodes } j_1, j_2,} \tag{2.11}$$

$$\boxed{b_k(t) = \lambda_7 a_k(t),} \tag{2.12}$$

$$\boxed{\theta_k(t) = \text{atan}(y_{j_1} - y_{j_2}, x_{j_1} - x_{j_2}),} \tag{2.13}$$

$$\boxed{\mathbf{x}_{c,k}(t) = \frac{1}{2}\left(\mathbf{x}_{j_1}(t) + \mathbf{x}_{j_2}(t)\right).} \tag{2.14}$$

Note that i and j index over node numbers, k over cells, and j_1 and j_2 denote the nodes that comprise cell k. Finally the petri-dish domain is given by

$$\boxed{\Omega = \left[-\frac{1}{2}L_{\text{petri-dish}}, \frac{1}{2}L_{\text{petri-dish}}\right] \times \left[-\frac{1}{2}L_{\text{petri-dish}}, \frac{1}{2}L_{\text{petri-dish}}\right] \subset \mathbb{R}^2.} \tag{2.15}$$

2.2 From qualitative to quantitative

The model developed thus far is a numerical framwork for simulating growing cell colonies. There is an element of randomness in the model: when two nodes are dislodged from the same point there is initially no preferred direction to move in so one must be chosen randomly before other forces can take effect. For this reason, every seperate run of the model starting with identical initial conditions will look very different after time has passed. It is expected however that some "averaged" quantity will stabilise so long as the model is simulated over a fairly large number of runs. Seperate runs of the model belong to a set called an ensemble. We will start with a relatively straightforward metric, the number of cells at time $T = 500$ steps.

2.3 Updating the nutrient field in time

In order to advance the nutrient field in time, an efficient and stable numerical solver was required. Because of the fact that the nutrient field $\hat{c}(\hat{x}, \hat{y}, \hat{t})$ is coupled to the biomass field which is dependent on a discrete network that changes in connectivity and node count, it was necessary to implement the method in-house, as opposed to using a pre-existing MATLAB boundary value solver for instance. The Crank-Nicolson method (Crank and Nicolson (1947)) was chosen for the practical reason that it led to faster simulations overall since a larger time step could be used as opposed to the explicit forward in time central in space (FTCS) scheme. This is a direct consequence of the fact that the (implicit) Crank-Nicholson scheme is unconditionally numerically stable. Let us begin by verifying this fact using the what is known as Von Neumann stability analysis (Charney et al. (1950)).

From this point we drop the hat on the PDE variables. Firstly, we define the nutrient field as $c^n_{j,k} = c(x_j, y_k, t_n)$ and similarly for $b^n_{j,k}$, the biomass. The values $c^n_{i,j}$ precisely satisfy the discretised equations which follow. Let $\bar{c}^n_{j,k}$ be the solution achieved with finite precison arithmetic operations which does not necessarily satisfy the equations, up to some error term,

$$\epsilon^n_{j,k} = \bar{c}^n_{j,k} - c^n_{j,k}.$$

Carrying out the Crank-Nicolson discretisation with the values $c^n_{j,k}$, we derive

$$\frac{c^{n+1}_{j,k} - c^n_{j,k}}{\Delta t} = \frac{\lambda_1}{2}\left(\nabla^2 c^{n+1}_{j,k} + \nabla^2 c^n_{j,k}\right) - \frac{1}{2} b^n_{j,k}(c^{n+1}_{j,k} + c^n_{j,k}),$$

$$\begin{aligned}\frac{c^{n+1}_{j,k} - c^n_{j,k}}{\Delta t} = {} & \frac{\lambda_1}{2}\left(\frac{c^{n+1}_{j+1,k} - 2c^{n+1}_{j,k} + c^{n+1}_{j-1,k}}{h^2} + \frac{c^{n+1}_{j,k+1} - 2c^{n+1}_{j,k} + c^{n+1}_{j,k-1}}{h^2}\right) \\ & + \frac{\lambda_1}{2}\left(\frac{c^{n}_{j+1,k} - 2c^{n}_{j,k} + c^{n}_{j-1,k}}{h^2} + \frac{c^{n}_{j,k+1} - 2c^{n}_{j,k} + c^{n}_{j,k-1}}{h^2}\right) \\ & - \frac{1}{2} b^n_{j,k}(c^{n+1}_{j,k} + c^n_{j,k}).\end{aligned}$$

To evaluate the numerical stability, we introduce a constant $\alpha = \frac{\lambda_1 \Delta t}{h^2}$. With this constant in place, we continue with

$$\begin{aligned}c^{n+1}_{j,k} - c^n_{j,k} = {} & \frac{\alpha}{2}\left(c^{n+1}_{j+1,k} + c^{n+1}_{j-1,k} + c^{n+1}_{i,j+1} + c^{n+1}_{j,k-1}\right) - 2\alpha c^{n+1}_{j,k} \\ & + \frac{\alpha}{2}\left(c^{n}_{j+1,k} + c^{n}_{j-1,k} + c^{n}_{j,k+1} + c^{n}_{j,k-1}\right) - 2\alpha c^{n}_{j,k} \\ & - \frac{\Delta t}{2} b^n_{j,k}(c^{n+1}_{j,k} + c^n_{j,k}).\end{aligned}$$

Since the discretised equations are linear in $c^n_{j,k}$ (taking $b^n_{j,k}$ to be slowly varying) and since $\bar{c}^n_{j,k}$ also form a solution, the $\epsilon^n_{j,k}$ are also a solution.

$$\begin{aligned}\epsilon^{n+1}_{j,k} - \epsilon^n_{j,k} &= \frac{\alpha}{2}\left(\epsilon^{n+1}_{j+1,k} + \epsilon^{n+1}_{j-1,k} + \epsilon^{n+1}_{j,k+1} + \epsilon^{n+1}_{j,k-1}\right) - 2\alpha\epsilon^{n+1}_{j,k} \\ &+ \frac{\alpha}{2}\left(\epsilon^{n}_{j+1,k} + \epsilon^{n}_{j-1,k} + \epsilon^{n}_{j,k+1} + \epsilon^{n}_{j,k-1}\right) - 2\alpha\epsilon^{n}_{j,k} \\ &- \frac{\Delta t}{2} b^n_{j,k}(\epsilon^{n+1}_{j,k} + \epsilon^n_{j,k}).\end{aligned}$$

We subsitute in an ansatz for the error term which is standard in Von Neumann stability analysis, namely

$$\epsilon^n_{j,k} = \xi^n e^{i\nu_x jh} e^{i\nu_y kh},$$

where ν_x and ν_y are wavenumbers and h is the spatial step. Note that $i = \sqrt{-1}$. In order for the numerical method to be stable, we require that $|\xi| \leq 1$ for all values of the wavenumbers ν_x, ν_y. Substituting this expression into the discretised equation for the error, we get,

$$\begin{aligned}\xi^{n+1}e^{i\nu_x jh}e^{i\nu_y kh} - \xi^n e^{i\nu_x jh}e^{i\nu_y kh} &= \frac{\alpha}{2}\left(\xi^{n+1}e^{i\nu_x (j+1)h}e^{i\nu_y kh} + \xi^{n+1}e^{i\nu_x (j-1)h}e^{i\nu_y kh}\right) \\ &+ \frac{\alpha}{2}\left(\xi^{n+1}e^{i\nu_x jh}e^{i\nu_y (k+1)h} + \xi^{n+1}e^{i\nu_x jh}e^{i\nu_y (k-1)h}\right) \\ &- 2\alpha\xi^{n+1}e^{i\nu_x jh}e^{i\nu_y kh} \\ &+ \frac{\alpha}{2}\left(\xi^{n}e^{i\nu_x (j+1)h}e^{i\nu_y kh} + \xi^{n}e^{i\nu_x (j-1)h}e^{i\nu_y kh}\right) \\ &+ \frac{\alpha}{2}\left(\xi^{n}e^{i\nu_x jh}e^{i\nu_y (k+1)h} + \xi^{n}e^{i\nu_x jh}e^{i\nu_y (k-1)h}\right) \\ &- 2\alpha\xi^{n}e^{i\nu_x jh}e^{i\nu_y kh} \\ &- \frac{\Delta t}{2} b^n_{j,k}(\xi^{n+1}e^{i\nu_x jh}e^{i\nu_y kh} + \xi^n e^{i\nu_x jh}e^{i\nu_y kh}).\end{aligned}$$

Dividing through by $\xi^n e^{i\nu_x jh}e^{i\nu_y kh}$, we attain,

$$\begin{aligned}\xi - 1 &= \frac{\alpha}{2}\left(\xi e^{i\nu_x h} + \xi e^{-i\nu_x h}\right) + \frac{\alpha}{2}\left(\xi e^{i\nu_y h} + \xi e^{-i\nu_y h}\right) - 2\xi\alpha \\ &+ \frac{\alpha}{2}\left(e^{i\nu_x h} + e^{-i\nu_x h}\right) + \frac{\alpha}{2}\left(e^{i\nu_y h} + e^{-i\nu_y h}\right) - 2\alpha - \frac{\Delta t}{2} b^n_{j,k}(\xi + 1).\end{aligned}$$

Using the fact that $\frac{e^{i\nu_x h} + e^{-i\nu_x h}}{2} = \cos(\nu_x h)$ and similarly for the other terms, we have,

$$\begin{aligned}\xi - 1 &= \alpha\xi\cos(\nu_x h) + \alpha\xi\cos(\nu_y h) - 2\alpha\xi \\ &+ \alpha\cos(\nu_x h) + \alpha\cos(\nu_y h) - 2\alpha - \frac{\Delta t}{2} b^n_{j,k}(\xi + 1).\end{aligned}$$

Finally, solving for ξ, and taking its absolute value we have,

$$|\xi| = \left| \frac{1 + \alpha \left[\cos(\nu_x h) + \cos(\nu_y h)\right] - 2\alpha - \frac{1}{2}\Delta t b_{j,k}^n}{1 - \alpha \left[\cos(\nu_x h) + \cos(\nu_y h)\right] + 2\alpha + \frac{1}{2}\Delta t b_{j,k}^n} \right| .$$

In order to complete the analysis, consider when biomass is $b_{j,k}^n = 0$. We have the simplified equation,

$$|\xi| = \left| \frac{1 - \alpha(2 - [\cos(\nu_x h) + \cos(\nu_y h)])}{1 + \alpha(2 - [\cos(\nu_x h) + \cos(\nu_y h)])} \right|,$$

Which has a numerator less than or equal to 1 because $\alpha > 0$ and $0 \leq 2 - [\cos(\nu_x h) + \cos(\nu_y h)] \leq 4$. The denominator is always greater than or equal to 1, which is enough to prove that the Crank-Nicolson scheme for the standard heat equation is numerically stable.

Whether this generalises to non-zero $b_{j,k}^n$ is determined by numerical experiments, and indeed for $\alpha = \frac{\lambda_1 \Delta t}{h^2} = \frac{(0.1)(0.01)}{(0.1)^2} = 0.1$, the scheme was not found to be unstable. However, for higher values of α it was possible to see spurious oscillations in the solution field for c which obviously suggests some form of instability.

2.4 Fast computational software in MATLAB

The software developed for the computational modelling is recquired to be fast, accurate and easy to understand.

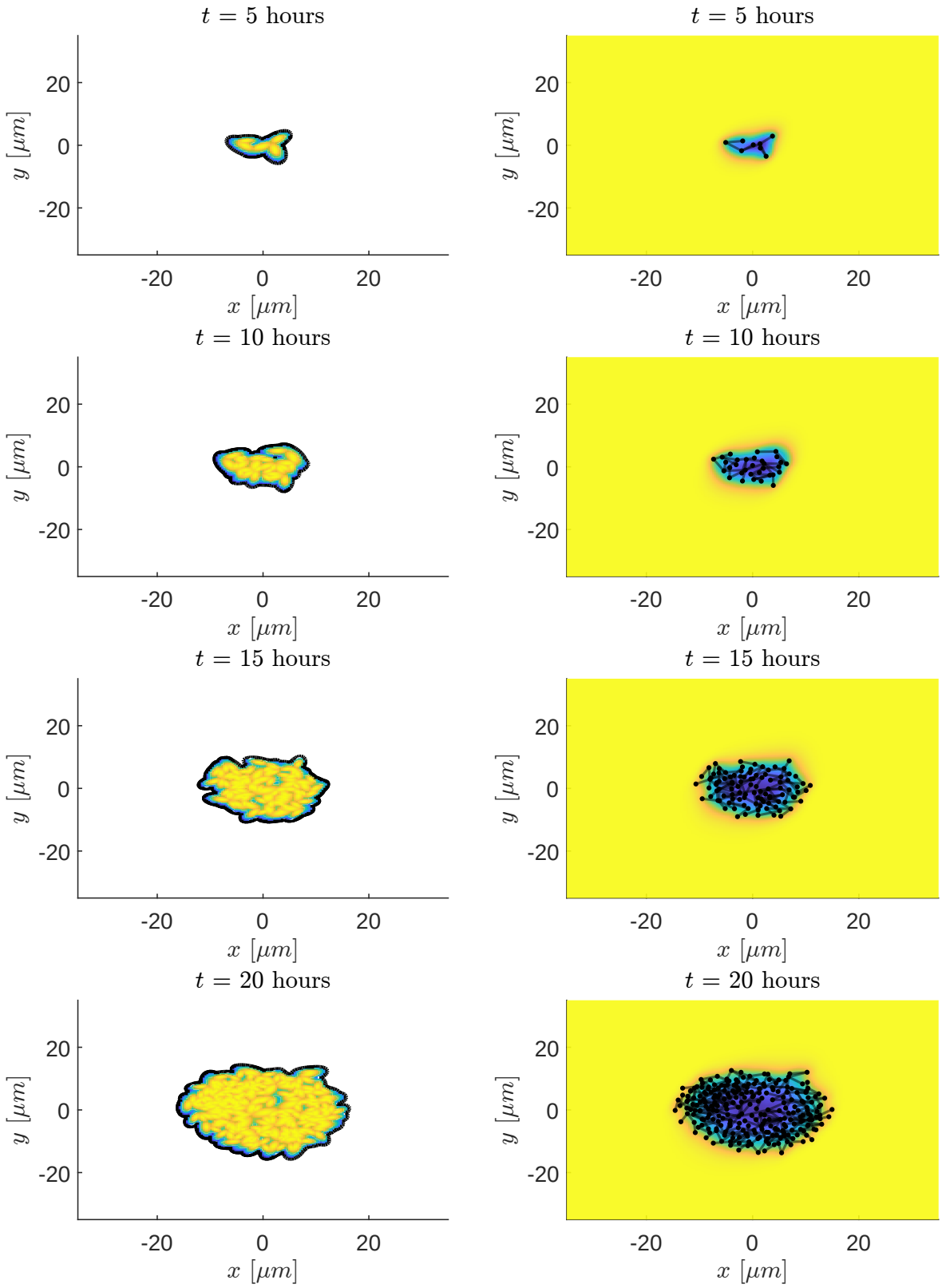

Figure 2.1: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 0.3$, $\lambda_7 = 0.5$. On the left we have the biomass field, the nutrient field is on the right.

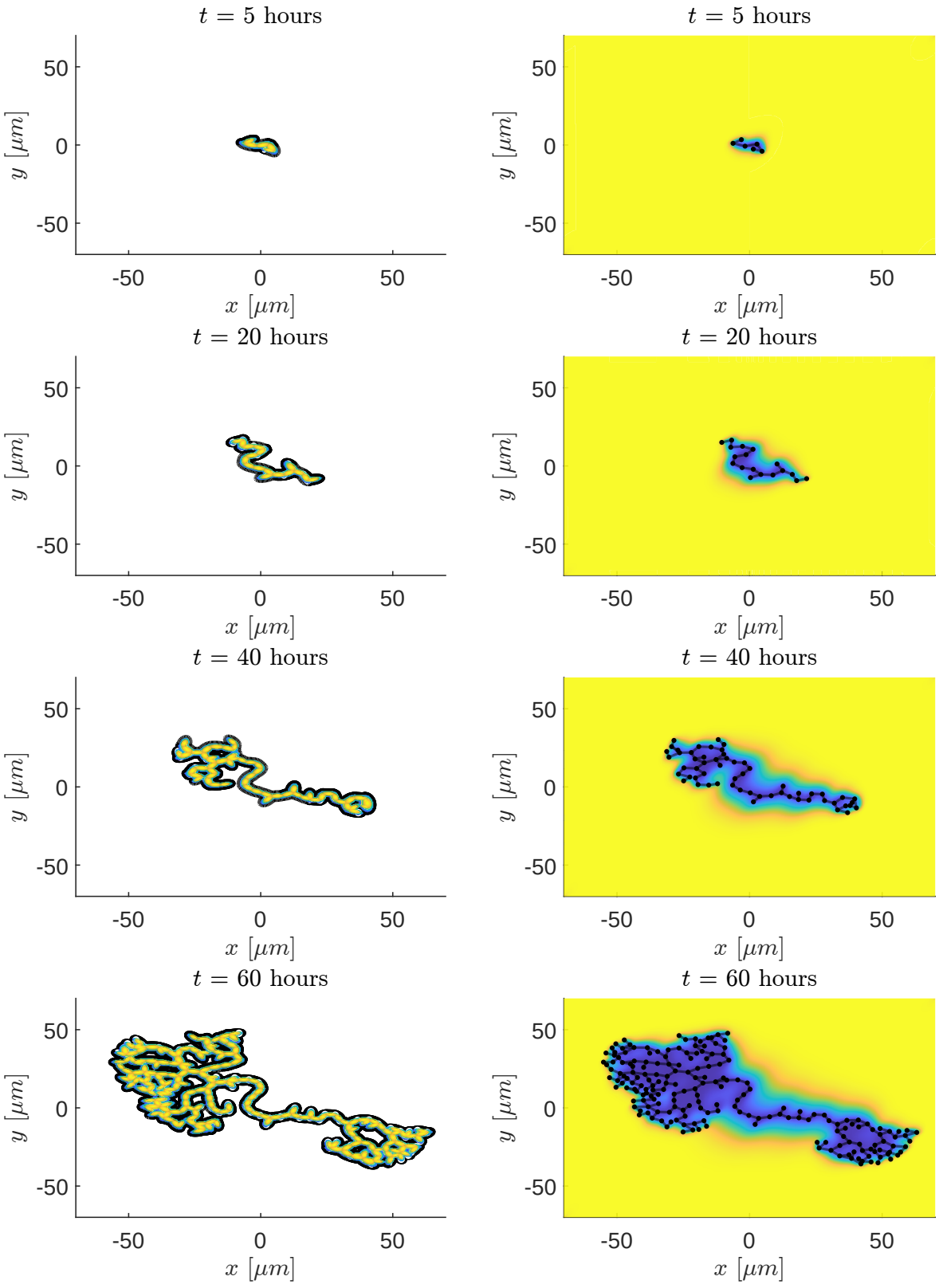

Figure 2.2: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 2.0$, $\lambda_7 = 0.5$. Biomass on left and nutrient field on the right.

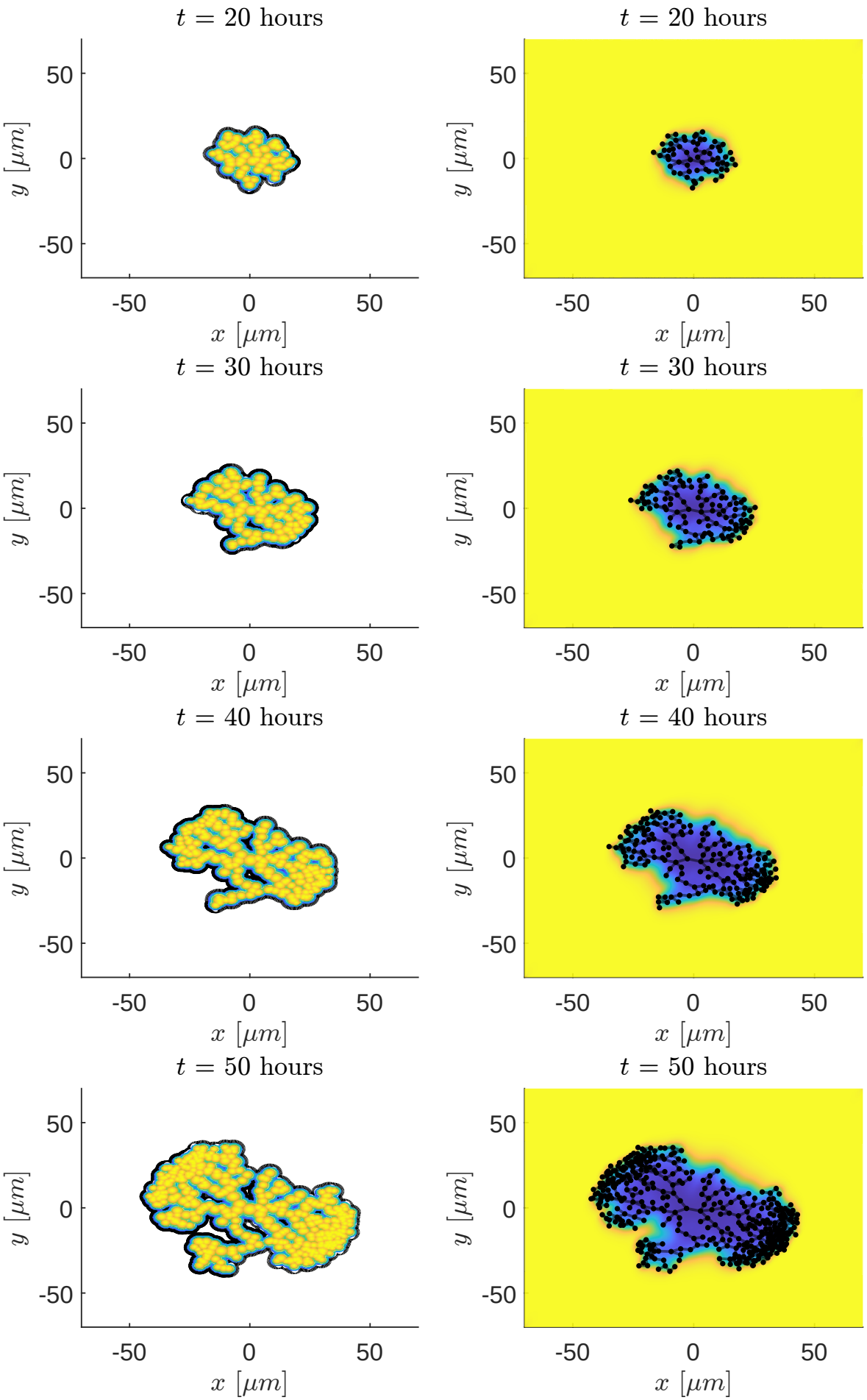

Figure 2.3: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$,

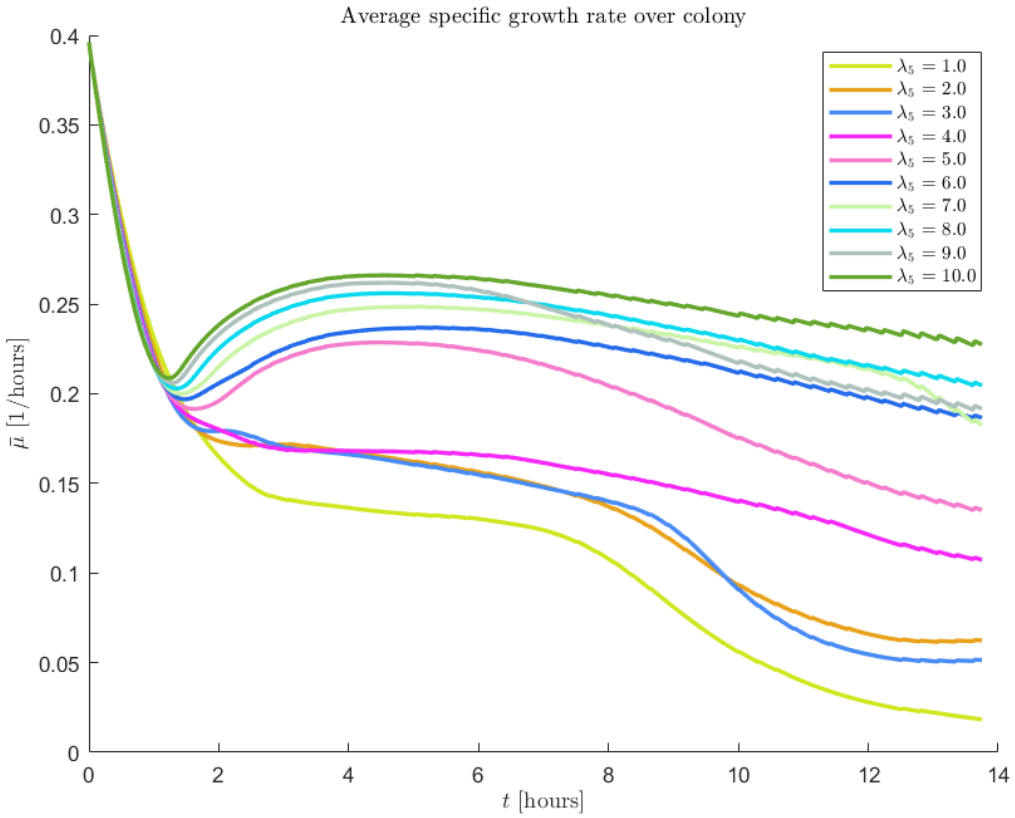

Figure 2.4: The colony average specific growth rate for different values of λ_5 is measured and plotted over time for ensemble size 1. The rest of the parameters took the values: $\lambda_1 = 0.1$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$, $\lambda_4 = 1.0$, λ_5 (variable), $\lambda_6 = 0.5$, $\lambda_7 = 0.7$. Remarkably, when $\lambda_5 \geq 5.0$ there is an interesting dynamic that occurs based on the compettition between nutrient consumption rate (λ_6) and the mobility (λ_5). For small values of mobility, the cells are not able to move enough into areas where the nutrient has not decayed.

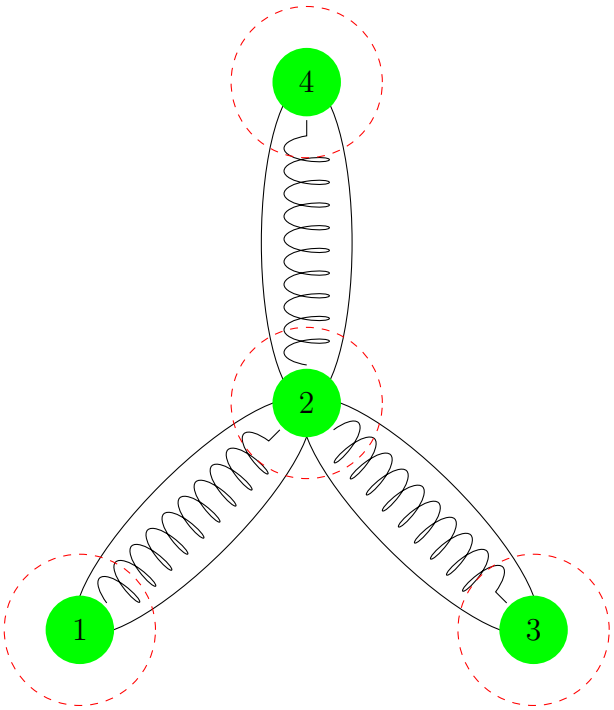

Figure 2.5: A diagram of three elliptical cells with internal springs to represent biomass elasticity ($\lambda_2 = \frac{K}{\eta\mu}$). The nodes are positioned at the ends of the major dimension of each ellipse and there are two nodes per cell. The force acting on node 2 for instance would be due to the forces from nodes 1, 3 and 4. The dashed red circles represent the activation radius ($\lambda_4 = R/L_0$) of the contact force between the nodes.

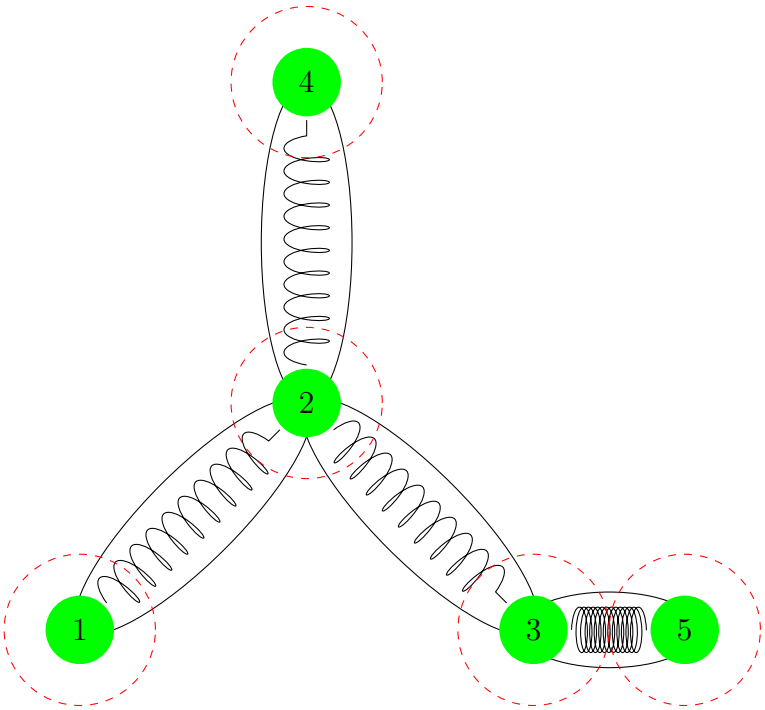

Figure 2.6: A mitosis event occurs via the addition of new nodes connected to old nodes. The nodes are added very close (distance $\delta \ll 1$) by the original nodes (exaggerated here) so that the spring force can be defined. After this point, the initially compressed cell "grows" outwards to achieve its nominal length, under the influence of elasticity, contact and chemotactic forces.

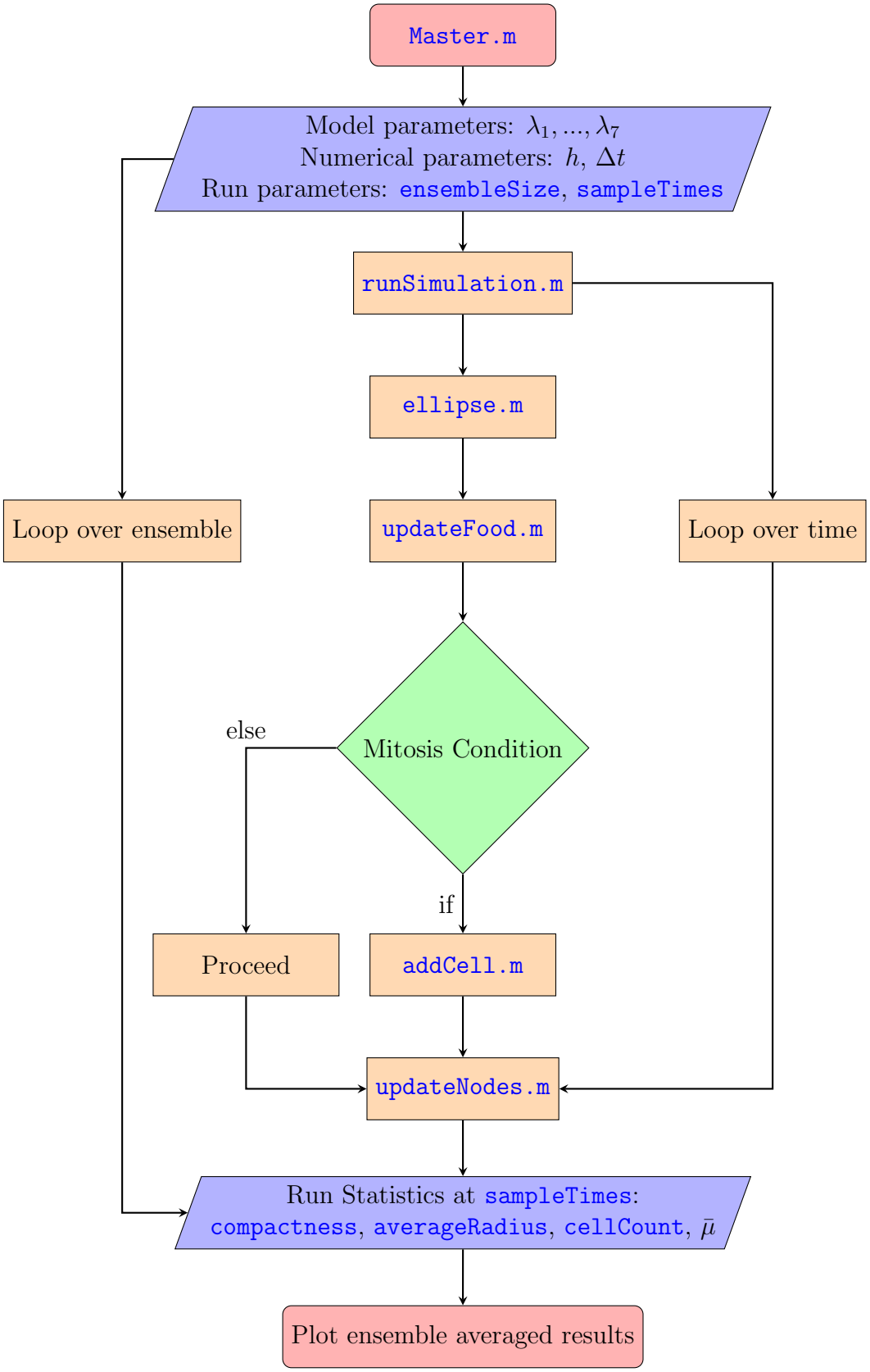

Figure 2.7: A flow chart of **CellColonySimulator** software package in MATLAB.

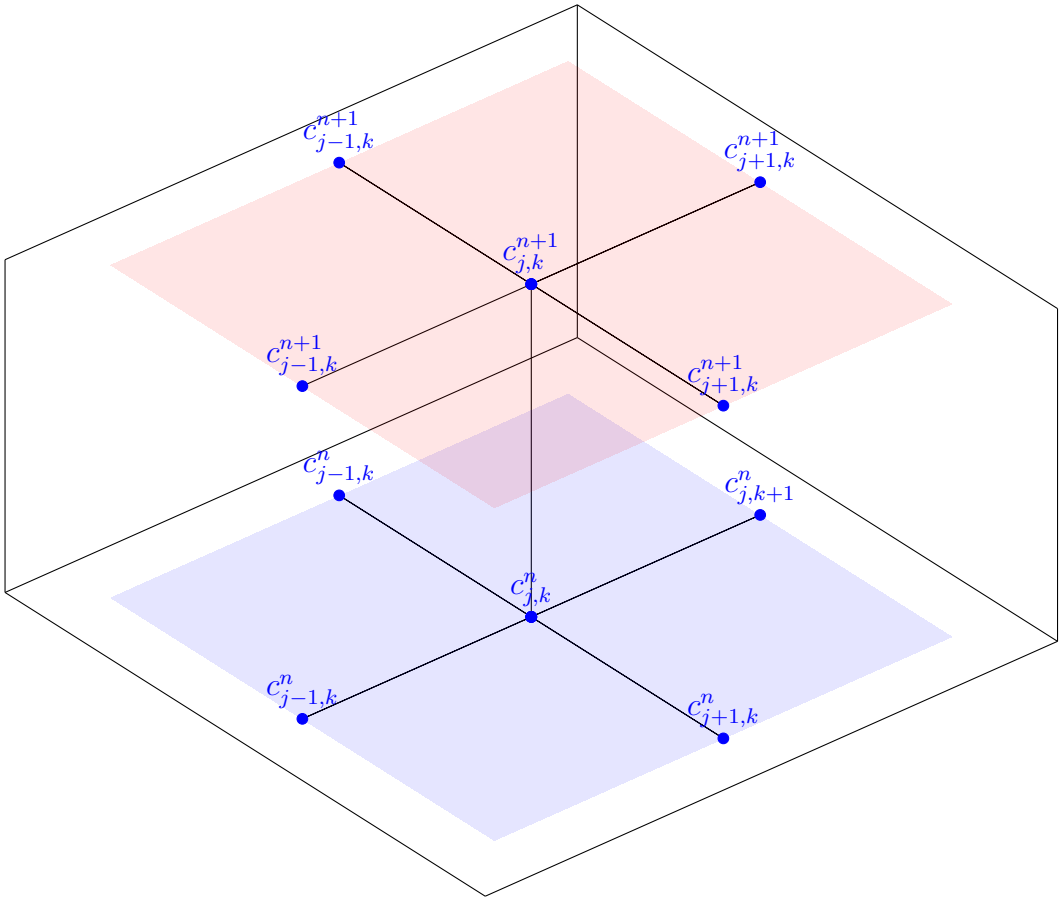

Figure 2.8: The five point stencil for the nutrient field at times $t_n = n\Delta t$ (blue plane) and $t_{n+1} = (n+1)\Delta t$ (red plane) used for the Crank-Nicolson scheme on the domain interior.

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 Witkin (1997) 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 θ_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 β-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 λ 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 λ 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 1000×1000 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.

Bibliography

Agnew, D., Green, J., Brown, T., Simpson, M., and Binder, B. (2014). Distinguishing between mechanisms of cell aggregation using pair-correlation functions. *Journal of Theoretical Biology*, 352:16–23.

Baraff, D. and Studios, P. A. Physically based modeling.

Borzov, S. (2021). *Use of Signed Distance Functions for the Definition of Protein Cartoon Representation*. PhD thesis, Masarykova Univerzit Brno, Czech Republic.

Charney, J. G., Fjörtoft, R., and Neumann, J. v. (1950). Numerical integration of the barotropic vorticity equation. *Tellus*, 2(4):237–254.

Crank, J. and Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical proceedings of the Cambridge philosophical society*, volume 43, pages 50–67. Cambridge University Press.

FUSEK, B. P. Visualization of sdf scenes.

Ghaffarizadeh, A., Heiland, R., Friedman, S. H., Mumenthaler, S. M., and Macklin, P. (2018). Physicell: An open source physics-based cell simulator for 3-d multicellular systems. *PLoS computational biology*, 14(2):e1005991.

Li, K., Green, J. E. F., Tronnolone, H., Tam, A. K., Black, A. J., Gardner, J. M., Sundstrom, J. F., Jiranek, V., and Binder, B. J. (2024). An off-lattice discrete model to characterise filamentous yeast colony morphology. *PLOS Computational Biology*, 20(11):e1012605.

Petrovic, L., Henne, M., and Anderson, J. (2005). Volumetric methods for simulation and rendering of hair. *Pixar Animation Studios*, 2(4):1–6.

Quilez, I. (2025). 2D distance functions. `https://iquilezles.org/articles/distfunctions2d/`.

Sauro, H. M., Hucka, M., Finney, A., Wellock, C., Bolouri, H., Doyle, J., and Kitano, H. (2003). Next generation simulation tools: the systems biology workbench and biospice integration. *Omics A Journal of Integrative Biology*, 7(4):355–372.

Turing, A. M. (1990). The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52:153–197.

Witkin, A. (1997). An introduction to physically based modeling: Constrained dynamics. *Robotics Institute*.