



A Novel Algebraic Agent Based Model for Pseudohyphal Baker's Yeast Colony Morphology

by

Isaac Nakone

In fulfilment of the requirements for the degree of

Bachelor of Mathematical Sciences - Honours

June 2025

School of Computer and Mathematical Sciences
Faculty of Sciences, Engineering and Technology

The University of Adelaide

Contents

Declaration	xi
Acknowledgements	xiii
Abstract	xv
1 Literature Review	1
1.1 An overview	1
1.2 Modelling Kingdom Fungi	1
1.3 Agent based modelling in context	2
1.4 From algorithmic botany to algorithmic mycology	4
1.5 Implicit surfaces and constructive solid geometry (CSG)	5
1.6 Algebraic topology meets mycology	6
1.6.1 Topologically associating domains (TADs) in genomics	6
1.6.2 A historical interlude	7
1.6.3 Grothendieck dreams	8
1.7 The soil underneath: partial differential equations (PDEs) for a nutrient source	9
2 Description of the model	11
2.1 Representing fungal cells algebraically	11
2.2 Cell colony dynamics with an underlying discrete network	15
2.3 The colony adjacency matrix, A	16
2.4 New cells from old	18
2.5 Incorporating the nutrient field	20
2.6 Non-dimensionalising the equations of motion	20
2.6.1 Non-dimensionalising the nutrient PDE	21
2.6.2 Non-dimensionalising the nodes ODE	22
2.6.3 Non-dimensionalising the biomass equation	23
2.6.4 Non-dimensional model equations of motion (EOMs)	24
2.7 Numerically solving for the nutrient field	25

3 Software Implementation: <i>CellColonySimulator</i>	29
3.1 The structure of the program	29
3.1.1 Calling the update loop in <i>runSimulation.m</i>	29
3.1.2 Vectorised computation of colony SDF in <i>ellipse.m</i>	32
3.1.3 A fast Crank-Nicolson scheme in <i>updateFood.m</i>	33
3.1.4 Introducing new nodes in <i>addCell.m</i>	35
3.1.5 Advancing the node positions using Euler's method in <i>updateNodes.m</i>	35
3.2 Statistics from the model	37
4 Numerical experiments and simulation results	39
4.1 First numerical experiment: <i>Compactness verus cell aspect ratio</i>	40
4.2 Second numerical experiment: <i>Growth rate versus mobility</i>	42
4.3 Summary: <i>Uncovering different growth regimes</i>	42
5 Discussion and limitations	57
5.1 Evaluation of the model	57
5.1.1 Conceptual limitations	57
5.1.2 Dynamical limitations	60
5.1.3 Computational consolations and cell death	62
5.2 Readiness for experimental validation and fitting	63
Conclusions	65
5.3 Summary remarks	65
5.4 A word on my scientific process	65
5.5 Philosophical remarks	66
Appendix	
A.1 Collocation algorithm with mask matrices	71
A.2 Cell-cell collisions with constrained dynamics	76
Acknowledging the use of generative AI	83
5.3 ChatGPT conversation on May 17, 2025: <i>Predicting criticism</i>	84
5.3.1 My input	84
5.3.2 ChatGPT output	84
5.4 ChatGPT conversation on May 17, 2025: <i>Philosophy of using algebraic curves in biology</i>	86
5.4.1 My input	86

5.4.2 ChatGPT output	86
--------------------------------	----

List of Tables

2.1	The variables associated with each cell indexed k and comprised of nodes i_1 and i_2	16
2.2	A summary of the numerical units	21
2.3	A summary of the dimensionless model parameters	23
3.1	The summary statistics extracted from the model	38
4.1	The rationale behind the selection of $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$	40

List of Figures

2.1	Two ellipses blended with various values of smoothness s . The bottom right figure is computed using standard minimum instead of smoothmin. Indeed, as $s \rightarrow 0$, the plot looks less smoothed across.	13
2.2	The separation d between two ellipse shapes with $s = 0.5$ and an aspect ratio of 0.9	14
2.3	Micrographs of a translucent yeast strain studied in [14]	15
2.4	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 blue circles represent the activation radius ($\lambda_4 = R/L_0$) of the contact force between the nodes.	16
2.5	A plot showing the underlying network of a simulated yeast colony as well as the level-0 contour of the corresponding colony SDF.	17
2.6	A mitosis event occurs via the addition of new nodes connected to old nodes. The nodes are added very close (distance $\delta \ll 1$) beside 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.	19
2.7	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.	26
3.1	A flow chart of CellColonySimulator software package in MATLAB.	30
3.2	A comparsion of high and low compactness	38
4.1	A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 3.0$, $\lambda_7 = 0.5$. On the left we have the biomass field, the nutrient field is on the right.	44

4.2	A cell colony with parameter values given by $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 0.1, \lambda_6 = 0.7, \lambda_7 = 0.73$. On the left we have the biomass field, the nutrient field is on the right. The aspect ratio is $\lambda_7 = 5.5/7.5 = 0.73$	45
4.3	A cell colony with parameter values given by $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.2, \lambda_6 = 0.7, \lambda_7 = 0.6$. Biomass on left and nutrient field on the right.	46
4.4	A cell colony with parameter values given by $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.6, \lambda_6 = 0.9, \lambda_7 = 0.95$. Biomass on left and nutrient field on the right.	47
4.5	A cell colony with parameter values given by $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.6, \lambda_6 = 0.7, \lambda_7 = 0.5$. Biomass on left and nutrient field on the right.	48
4.6	The colony compactness and growth rate for $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.0, \lambda_6 = 1.0, \lambda_7 = 0.4, 0.5, 0.7, 1.0$ and an ensemble size of 6.	49
4.7	The average colony compactness for $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.0, \lambda_6 = 1.0, \lambda_7 = 0.4, 0.5, 0.7, 1.0$ and an ensemble size of 6.	50
4.8	Compactness for ensemble instance $e = 1$ and $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.0, \lambda_6 = 1.0, \lambda_7 = 1.0$. The inset plots show the biomass at three times for comparison with compactness.	51
4.9	Compactness for ensemble instance $e = 1$ and $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 1.0, \lambda_6 = 1.0, \lambda_7 = 0.5$. The inset plots show the biomass at three times for comparison with compactness.	52
4.10	Specific growth rate $\mu(t)$ for ensemble instance $e = 1$ and $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 5.0, \lambda_6 = 1.0, \lambda_7 = 0.7$	53
4.11	Specific growth rate $\mu(t)$ for ensemble instance $e = 1$ and $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 0.1, \lambda_6 = 1.0, \lambda_7 = 0.7$	54
4.12	Compactness $C(t)$ and specific growth rate $\mu(t)$ for $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 0.1, 1.0, 2.5, 5.0, \lambda_6 = 1.0, \lambda_7 = 0.7$, and an ensemble of size 6. Note that the bottom panels could only be simulated to $t = 20$ hours due to GPU out-of-memory issues.	55
4.13	Average growth rate $\mu(t)$ for $\lambda_1 = 0.1, \lambda_2 = 5.0, \lambda_3 = 5.0, \lambda_4 = 0.5, \lambda_5 = 0.1, 1.0, 2.5, 5.0, \lambda_6 = 1.0, \lambda_7 = 0.7$, and an ensemble of size 6. Note the plot for $\lambda_5 = 5.0$ could not be completed on the available hardware due to GPU out-of-memory.	56

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

19/6/2025

Acknowledgements

Where to begin?

Thank you to my supervisors, Ed Green and Sanjeeva Balasuriya, for your enduring support during these *two and a half* years.

Thankyou to my highschool English teacher, Tanja, for teaching me how to use form and technique in my writing.

Thank you to my parents, Debbie and Alex, and to my sister, Amelia. In your own way, you each provided the soil from which this thesis grew.

Thank you to my literary friend, John, whose cerebral writing will have inspired a generation of *rebels with a cause* by next Tuesday, I'm sure of it.

Thank you to Kai and my other colleagues at *Adelaide University* for providing the environmental stresses to keep me growing. Ki was always willing to speak with me and offer his time. I am grateful.

There appears to be a collection of *Alex*'s in my life. I thank you all dearly in one foul sweep.

To *Radee*, for reminding be how to thinking critically and creatively at the same time. To Harry and Zehao and the other engineers one of whom is another *Alex*, I appreciate you.

To Giuseppe, thank you for introducing me to the *bible of statistical mechanics*.

For the fruitful *Kong-Tchorbadjiev-Nakone* (KTN) correspondence, for your years of philosophical musing.

Thank you to *cafe* staff at the university for your unassuming presence among the day-to-day emergence of new forms.

Thankyou to the late philosophy Professor Michael Sugrue, whose lectures, both young and old, were stunning depictions of the daring heart of philosophy.

Thankyou to *ChatGPT*, for helping me externalise my interior geometry. It materialised in the form of the great parable of baker's yeast.

If there was anyone I forgot, I hope you are not offended.

Isaac

Abstract

Baker's yeast (*Saccharomyces cerevisiae*) is capable of undergoing a pseudohyphal transition to filamentous growth in nutrient poor environments. An algebraic representation of *S. cerevisiae* cells is adapted from computer graphics to model cellular division (mitosis) at low computational cost. By reimagining mitosis as topological bifurcation using level sections of signed distance functions (SDFs) underpinned by a discrete mechanistic network, novel predictions about *S. cerevisiae* colony morphology are made. Coupling the algebraic biomass field to a traditional reaction diffusion system for nutrient allows the relationship between metabolism and growth rate to be estimated for this agent based model (ABM). A measure for filamentous branching called compactness is introduced and related to cellular mobility under chemotaxis through numerical experiments.

Chapter 1

Literature Review

1.1 An overview

This thesis models *Saccharomyces cerevisiae* (baker’s yeast), using algebraic curves coupled with a nutrient partial differential equation. A review of the literature is supplied to motivate the modelling choices from a range of perspectives.

1.2 Modelling Kingdom Fungi

Microfossils discovered in the Grassy Bay Formation (Shaler Supergroup, Artic Canada) dating back to the Proterozoic Eon, 2500 to 538.8 million years ago, were found to have fungal affinity [30]. Curiously, it took until 1969 for Fungi to be recognised in plant ecologist Robert Whittaker’s work as part of a taxonomic kingdom, Kingdom Fungi, which is distinct from Kingdom Plantae [47]. Today, locating fungal species in phylogenetic lineages is facilitated through colonial morphology imaging as a first estimate. More advanced experimental techniques such as transmission electron microscopy (TEM) can provide conclusive identification [30] of fungal samples.

Morphological features observed in fungal strains such as fission yeast (*Schizosaccharomyces pombe*) are sometimes cited as the reason why yeast is broadly considered a “model organism” [23]. Cell Biologist Murdoch Mitchison was able to study the fission yeast cell cycle in 1957 for instance, because the length of the rod shaped cells of *S. pombe* reflected their stage in the cell cycle. It happens to be an *S. pombe* spore grown on malt extract agar that was imaged in exquisite detail via X-ray diffraction microscopy by [25]. By 3D reconstruction techniques [25] visualised the organelles present in the unstained (and unsectioned) spore at a resolution of 50 – 60 nm producing isosurfaces of cellular organelles such as the endoplasmic reticulum, vacuole and cell wall.

Contemporary advancements in computer hardware facilitate the ability for big datasets such as 3D morphological data obtained from yeast to be analysed mathematically. [9] discuss the “inverse problem” in mathematical biology, which is to determine a model from noisy, complex and extensive biological data whilst minimising bias. This often comes in the framework of model parameter estimation which can be accelerated using fairly systematic deep learning techniques. Still, the predictive power of a mathematical model is always limited by the bias implicit (or explicit) in its assumptions.

Biology is inherently multi-scale and the problem of bridging between scales is named the “translational challenge” [1]. The interplay between the phenotypical properties of a filamentous yeast colony and its genotype, is one such challenge. Indeed, the eukaryote *S. pombe*, has a similar cytoskeletal organisation to a human tissue cell as well as analogous DNA transcription mechanisms [24] and yet human tissue morphology vastly differs from fission yeast.

[1] position agent-based models (ABMs), dating back to cellular automata, as well suited to the translational challenge for a number of reasons. These include the ability for ABMs to model morphological emergence from simple and modular rules. These rules are constructed from a qualitative understanding of the underlying mechanisms of for instance *S. pombe* intracellular stresses which could be based on some basic assumptions about cellular elasticity. Importantly, ABMs are easy to deploy acting as a cheap alternative to *in vitro* experiments with fungal species.

Multi-cellular agent based modelling of *S. pombe* or *Saccharomyces cerevisiae* (baker’s yeast) is of interest to a number of scientific communities. On the one hand, due to the molecular similarity of yeast to human cells, it can serve as a model organism that can ethically be grown, and on the other, it is of intrinsic interest to phylogeneticists who study the origin of life on Earth. Morphological studies using ABMs can also provide breweries and food processing industries with practically useful findings to set up ideal growth conditions for cultures prior to expensive industrial implementations. Indeed *pombe* is Swahili for “booze”, being originally observed in contaminated millet beer delayed on route from East Africa to Germany [23].

1.3 Agent based modelling in context

In the current thesis, a dynamic off-lattice agent based modelling framework for *S. cerevisiae* colonial morphology is proposed and implemented. As opposed to fission yeast, *S. cerevisiae* exhibits budding and can be induced into a filamentous regime

called pseudohyphal growth in which cells become elongated leading to overall branching patterns in the colonial morphology. The off-lattice ABM developed in [28] inspired the modelling of the pseudohyphal *S. cerevisiae* regime presented here, though it uses a vastly different computational methodology.

[28] use approximate bayesian computatation (ABC) to infer model parameters within the context of a probabilistic (and spatial) branching simulation, yielding remarkable fits to micrographs of filamentous yeast. Building from the experimental observations of pseudohyphal growth presented in [20], Li et al. dispense with diffusion limited growth (DLG) which is shown by [43] to be just a peripheral mechanism in filamentous colony shape.

Whilst pseudohyphal growth dynamics have proved to be the dominant growth mechanism [43] for *S. cerevisiae* in nutrient poor environments, it is the aim of the present work to reintroduce a diffusive nutrient medium in the hope of defining *to what extent* these two phenomena intertwine.

[6] develop rigid body modelling for multi-cellular colonies using an ABM which represents each cell as a deformable polygon. The emphasis in the model of Brown et al. is to represent the process of cell division in diverse biological settings through the addition of new nodes and edges to each polygonal cell. In contrast, [28], use static ellipsoidal cells that are added into the simulation abruptly. [28] instead focus on the ability to fit their model to experimental micrographs of *S. cerevisiae* using statistical methods. Interestingly, this choice seems to reflect a gap in the multi-cellular modelling literature: *how can cell division be represented without the artificial inclusion of new parameters?*

In both the work of [46], and the ABM of [6], the dynamics of the colony are derived from the dynamics of individual cells which are represented by meshes with time-dependent topology as well as geometry. However ingenious this idea may be, computation quickly becomes intractible for large colonies due to the increasing size of the underlying system of ordinary differential equation (ODEs) for the node states. Most ABMs need supercomputers, however the computationally costly use of meshed cells is circumvented in the current work by employing an algebraic representation of cell geometry.

1.4 From algorithmic botany to algorithmic mycology

The model defined in Chapter 2 of this thesis is underpinned by a “growing network” which is an example of a graph-rewriting system. The natural precursor to this concept is the mathematical corpus of biologist Aristid Lindenmayer, whose work on cellular interactions in plants and fungi based on *sequential machines*, [29], developed into the notion of an *L*-system, [39]. An *L*-system is a string-rewriting system that applies a conversion rule to each letter in a word simultaneously, [39]. For example, the rules (or “productions”) that sends $A \rightarrow AB$ and $B \rightarrow A$, would evolve the input word, A , as follows,

$$\begin{aligned} &A \\ &AB \\ &ABA \\ &ABAAB \\ &ABAABABA, \end{aligned}$$

which could continue on indefinitely. Taken together, the alphabet, $\{A, B\}$, the productions, $\{A \rightarrow AB, B \rightarrow A\}$, and the intial word, A , comprise an *L*-system.

The ideas of Lindenmayer have been taken up by the Algorithmic Botany Group [38] led by Przemyslaw Prusinkiewicz whose website has links to the group’s papers dating back to the 1980s. In a recent paper from 2024, Coen and Prusinkiewicz study two modes of developmental timing mechanisms in plants, one of them being related to intrinsic molecular processes and one of them being growth-dependent, for instance, dependent on space being available [10]. Their computational methodology, which is of relevance to the current thesis, seems to be based on novel *L*-system softwares developed in the Algorithmic Botany Group. Even by the year 2012, Boudon et al. [5] published *L-Py*, a Python package for modelling plant architecture based on a dynamic language.

The key innovation of the present thesis work is to bring into contact three distinct areas of study into one interdependent framework for studying mycology. These three areas are,

- Graph rewriting as exemplified in botanical *L*-system work,
- Constructive solid geometry (CSG) in computer vision,
- Reaction-diffusion partial differential equations (PDEs), as studied in mathematical biology [44].

The first and third of these areas have already been mentioned here earlier when referring to the work of the algorithmic botany group and the recent paper from [43]. The following section addresses the second dot point.

1.5 Implicit surfaces and constructive solid geometry (CSG)

The three most common ways of representing a 2D or 3D object mathematically are parametric, polygonal, or implicit techniques. For instance, a parametric representation of the points of a circle is $(\cos t, \sin t)$ where t is a real number. Alternatively, a polygonal representation of a circle would be a regular n -gon which could be made a better approximation as $n \rightarrow \infty$. This sort of approach is effectively used by [46] and [6] in their solid mechanics frameworks. Mesh based modelling has the advantage of locality in that deformations to the shape can be made by moving a node in the mesh without changing it globally which gives more flexibility. A limitation which has already been mentioned is that mesh based models, particularly in the context of growth, quickly lead to huge node numbers which imply the necessity of supercomputers for large colony simulations.

Conversely, implicit representations of smooth geometric shapes are defined globally at the cost of local specificity. In three dimensions, one can define the surface of a sphere as set of points in \mathbb{R}^3 which satisfy the following equation,

$$x^2 + y^2 + z^2 - 1 = 0.$$

This is a special case of what is known as a quadric surface, which is defined as the set of points which satisfy $f(x, y, z) = 0$, where

$$f(x, y, z) = p_1x^2 + p_2xy + p_3xz + p_4x + p_5y^2 + p_6yz + p_7y + p_8z^2 + p_9z + p_{10},$$

as introduced in [4]. The full generalisation of implicit curves would require a precise notion of what space the functions, f , belong to. In the case of quadric surfaces, the functions belong to the ring of polynomials in x , y and z , written as $\mathbb{R}[x, y, z]$, but more exotic settings can be used.

In many contexts where specificity and local shape control are essential, implicit curves are not the best tool. Consider the quadric surface defined above. The only way to modify the surface is to tune the parameters p_1, \dots, p_{10} but doing this results in a change to the whole surface which requires advanced mathematical machinery to control. This is where constructive solid geometry (CSG) comes in.

CSG is a technique used to build new implicitly represented shapes from old ones. In multi-cellular modelling, where the individual cell geometry is more or less fixed but there are regular topological bifurcations corresponding to cell division, CSG is ideal and surprisingly has not been implemented in mycology agent based models as far as the author is aware. That being said, a recent paper from 2023 models the time-dependence of cells of bristle worms (*Platynereis dumerilii*) using signed distance fields (SDFs) which are learnt from biomedical images using neural networks, [48]. As early as 2019, one could learn the continuous SDF of arbitrary 3D geometry using *DeepSDF* created by [36], which employs neural networks to learn from noisy input data. [48] improve on *DeepSDF* by using *sine* activation functions and more hidden layers in their multi-layer perceptron (MLP) neural network. Whilst [48] produce time-dependent implicit cell representations using an impressive data-driven approach, this thesis focuses on developing a mechanistic model that can bridge the gap between the cell and colony scales using implicit curves.

The graphical technique used in the current thesis has its origin in “Metaballs” or “Blobby modelling” which is in one case implemented in C++ by [27]. The individual balls, modelled by quadric surfaces f_1 and f_2 , are blended together into a new quadric surface $f(x, y, z) = f_1(x, y, z) + f_2(x, y, z)$. The idea then is to construct complex but smooth geometry from smaller algebraic components.

1.6 Algebraic topology meets mycology

1.6.1 Topologically associating domains (TADs) in genomics

By 1997, [22] and [32], had independently discovered *cohesins*, proteins that prevent “premature splitting” of sister chromatids due to microtubules, facilitating the correct timing of mitosis. Both teams used *S. cerevisiae* cells and both used *Fluorescence In Situ Hybridization* (FISH) in order to visualise the budding yeast genome. Dekker et al. (2002) invented a higher resolution genomic imaging technique called 3C (chromosome conformation capture) [12] providing a less invasive alternative to FISH, and using *S. cerevisiae* for this comparison. In 3C, formaldehyde fixation is used to cross-link nearby proteins. Relative frequencies at which distinct base-pairs (from different proteins) interact over the genome are counted by a sophisticated experimental process known as “ligation” which involves a polymerase chain reaction (PCR), [12].

The 3C, 4C (chromosome conformation capture-on-chip/circular chromosome conformation capture) and 5C (chromosome conformation capture carbon-copy) techniques have culminated in Hi-C which uses a combination of 3C and next generation

sequencing (NGS) to image “higher order” interactions in the cell nucleus, [33]. Another pair of teams, [33] and [13], identified topologically associating domains (TADs) in 2012. They imaged mouse embryonic stem cell genomes using 5C.

As indicated by the “TAD caller”, de Wit (2019), weakly interacting regions of genome content that interact strongly within themselves, called TADs (somewhat provisionally) evade clear definition, [49]. TADs are an interpretation of the diagonal blocks on a Hi-C plot, a matrix of genomic content proximities, [49]. It is worth noting that there are other Hi-C structures (blocks) that have inspired new names: “loops” or “compartments”, “subTADs”, “microTADs”, [3]. In 2016, [16] identified TADs in baker’s yeast. The question emerges: *is any subcomponent of the living (fungal) cell essential to its growth?* De Wit (2019) suggests that “a more constructive path will be to explain TADs in light of the mechanisms that form them, rather than describing TADs as we see them”.

1.6.2 A historical interlude

A note: In this subsection I comment on some of the ethical dimensions of mathematical modelling in biology, touching on the confronting subject of eugenics. I hope that by facing such dehumanising content in a scientific context we can identify the way mathematical forms have been historically shaped by ideology.

Sir Ronald Alymer Fisher, whose ashes are buried in St. Peter’s cathedral in Adelaide, Australia, declared in 1959 that more “attention to the History of Science ... by biologists is needed”, as quoted by [15] in an anecdotal commentary. Edwards, A (2008) cites a selection of interesting anecdotes from Godfrey Harold Hardy and Wilhelm Weinberg who independently discovered the Hardy-Weinberg equilibrium and were both met by Fisher. Fisher, who discovered the statistical technique called ANOVA, for analysis of variance, in [18] (1919), quoted “a considerable body of pedigree evidence” for a “single mendelian factor” cabable of producing “feebleness of mind”, [17]. Fisher attempts to refute Reginald Punnett’s, “anti-eugenic propoganda”, [17], by critiquing Punnett’s data table on the number of generations required to decrease the proportion of “defectives”. He substitutes another table of 325 “feebleminded cases”, including “alcoholic, sexually immoral, criminalistic, ... insane, ... tramps”. It seems making East African *pombe* is a more noble scientific cause.

R. A. Fisher found creative ways to propogate his ideas. For example, Fisher, independently of Andrey Kolmogorov, Ivan Petrovsky, and Nikolai Piskunov, developed a partial differential equation (PDE) known as the KPP-Fisher or Fisher-kolmogorov equation, [19]. At the *Trinity High Table*, Hardy is paraphrased (by Edwards, A, 2008) to have said that “if a system of axioms form the deduction of a contradiction,

then any proposition can be deduced from it”. The actual quote is from Fisher’s 1958 anecdote [15], in which the implication that “McTaggart and the Pope are one”, could be deduced if the contradiction “two equal to one” could first be derived. Nowadays, new mathematical machinery has emerged which treat such matters with more nuance.

1.6.3 Grothendieck dreams

Note: In this subsection, I name my mathematical role-model and hero, Alexander Grothendieck, who I believe exemplifies the living spirit of mathematics.

Grothendieck is quoted in [31] to have described his mathematical process through analogy.

“The first analogy that came to my mind is of immersing the nut in some softening liquid, and why not simply water? From time to time you rub so the liquid penetrates better, and otherwise you let time pass. The shell becomes more flexible through weeks and months—when the time is ripe, hand pressure is enough, the shell opens like a perfectly ripened avocado!”

The question bubbles up: *Does Grothendieck’s yoga of forms, his pure mathematical methodology, transfer over to applied mathematics, or even mathematical biology?* Is not the mechanism by which *cohesin* detaches from the cell’s sister chromatids during *anaphase* allowing for the splitting of the spindle apparatus, an example of patience and precise timing?

In this literature review, I have introduced some of the technical and ethical dimensions of mathematical modelling in biology with a focus on *S. cerevisiae*. If topologically associated domains (TADs) are to be taken seriously as a subject of foundational mathematical study then it seems conceivable that the whole spatial arrangement of a cell could be viewed as a TAD in itself, perhaps a “superTAD” to denote that it contains TADs, subTADs, microTADs and so on.

I currently lack the expertise to formalise such a theory in a precise manner. That being said, it is interesting to consider whether the *whole* cell can be thought of as a topological space. A physical yeast cell on a petri-dish is a messy and imperfect entity which appears to be the *furthest* thing from a clean mathematical object, both in shape and function. With the advent of tools like *DeepSDF*, and topological data analysis (TDA), a clean mathematical description of *best fit* can be derived, even if the cell itself is a site of living texture.

The question of *biological laws* then becomes: *can we predict the best-mathematical-fit for a biological assemblage, even if we cannot predict (or even represent completely) the actual cell?* My hunch is that a simple “form of best fit” really does emerge, and with that idea taken on a reasoned faith I proceed. Somewhat playfully, I supply the metaphysical interpretation that there is no single factor: *Life’s essence is the whole.*

I represent cells as level-sections of polynomials in two spatial variables x , and y . By *level-section* I mean the set of points $(x, y) \in \mathbb{R}^2$ that satisfy,

$$f(x, y) \geq 0,$$

where $f(x, y)$ is a particular member of the ring of polynomials, $\mathbb{R}[x, y]$. In the case of an ellipse which is the polynomial,

$$f(x, y) = 1 - \left[\left(\frac{x}{p} \right)^2 + \left(\frac{y}{q} \right)^2 \right],$$

where $p, q \in \mathbb{R}$, there is a point of vanishing gradient at the origin. I call that point the center of the ellipse, which could be translated to new position, (x_0, y_0) , via the change of variables $x \mapsto x - x_0, y \mapsto y - y_0$. In Chapter 2, the mechanism of cell division is modelled by combining polynomials of this form into a new one using MATLAB’s `min`. In section 1.7, I introduce the modelling of the diffusive nutrient.

1.7 The soil underneath: partial differential equations (PDEs) for a nutrient source

A reaction-diffusion PDE is employed to model the underlying nutrient field, $c(x, y, t)$. The yeast colony biomass given by the field, $b(x, y, t)$ (determined by the colony’s algebraic representation) interacts with c by a mechanism of aggregate gradient sensing over long periods of time [26]. As per Keller and Segel’s *Model for Chemotaxis*, I represent the external force applied to the biomass, by the nutrient field’s spatial gradient ∇c , [26].

The nutrient field is affected by the presence of the biomass, and this is summarised by the reaction-diffusion PDE, [19],

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) - rbc,$$

where the second spatial derivatives are called the diffusion term, and the additional expression, $-rbc$, is a reaction between the biomass and nutrient at rate, r . If b is

large it means the yeast is more dense so the rate of nutrient depletion is larger. But c must be a coefficient to prevent the nutrient dropping to unphysical negative numbers.

I finish this literature review with a wonderful insight from the great D'Arcy Wentworth Thompson's *On Growth and Form*, [42],

"Still, all the while, like warp and woof, mechanism and teleology are interwoven together, and we must not cleave to the one nor despise the other; for their union is rooted in the very nature of totality."

Chapter 2

Description of the model

2.1 Representing fungal cells algebraically

The use of signed distance fields (SDFs) to model organic surfaces is a time honoured graphical technique used, for example, by Pixar Animation Studios to model hair in *The Incredibles* (see [37]). 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 [40]. In this thesis, we do not use signed distance fields based on the practical reason that their definitions usually involve square roots which are slow to compute. We employ very similar formula without the square roots, which yield the same level sets (in 2D) but are fast to compute.

To motivate the primary mechanism by which cells will undergo mitosis in this thesis, we consider a toy example in which the level sections of the equations for a pair of ellipses undergo a catastrophic topological change as one real parameter changes, namely the distance d between their centers. We start by considering the equations for two ellipses which begin as coincident and move apart as the parameter d becomes larger. The equations that represent the individual cells are

$$f_1(x, y) = \left(\frac{x+d}{p}\right)^2 + \left(\frac{y}{q}\right)^2 - 1,$$

$$f_2(x, y) = \left(\frac{x-d}{p}\right)^2 + \left(\frac{y}{q}\right)^2 - 1,$$

where $p = 1.0L_0$ and $q = 0.9L_0$ are the semi-major radius and semi-minor radius of the ellipses, respectively, and d is the distance between their centers. The aspect ratio is $q/p = 0.9$ and $L_0 = 7.5 \mu\text{m}$ is the average length of a baker's yeast cell in ideal conditions as per [8].

In order to generate the combined algebraic curve for the pair, which, by an abuse of notation, will be called the colony SDF from now on, we build the following. This is done using what is called a “union” in computer graphics. This is simply the pointwise minimum,

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

To get a smooth transition between the cells as they come apart we could alternatively use the smoothmin which is defined by a smoothness parameter s as in

$$\text{smoothmin}(f_1(x, y), f_2(x, y); s) = -s \log(e^{-f_1(x, y)/s} + e^{-f_2(x, y)/s}).$$

Figure 2.1 shows the effect of using different values of smoothness s . Ultimately, using smoothmin was abandoned in favour of the computationally quicker `min`, which does not require the costly evaluation of the natural log and exponential functions.

As shown in figure 2.2, we have a smooth splitting of a cell as the parameter d ranges from 0.0 to $9.1 \mu\text{m}$. Here s is the smoothing parameter. In order to ensure that only the interior of the SDF level section was plotted, I set positive entries to `nan`, which is MATLAB code for *not a number*. The surface plots in figure 2.2 were produced via the use of MATLAB's `surf` function which ignores `nan` entries in the underlying height matrix. One additional transformation made before plotting, was to multiply by -1 to reflect the SDF about the level plane, achieving a positive value inside the biomass.

A signed distance field for an ellipse is used to model baker's yeast cells which are in a pseudo-hyphal growth regime. An ellipse centered at the origin with semi-major dimension p (the x intercept) and semi-minor dimension q (the y intercept) has an SDF given by

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

Recall that for computational efficiency we choose to use,

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

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. We can also translate and

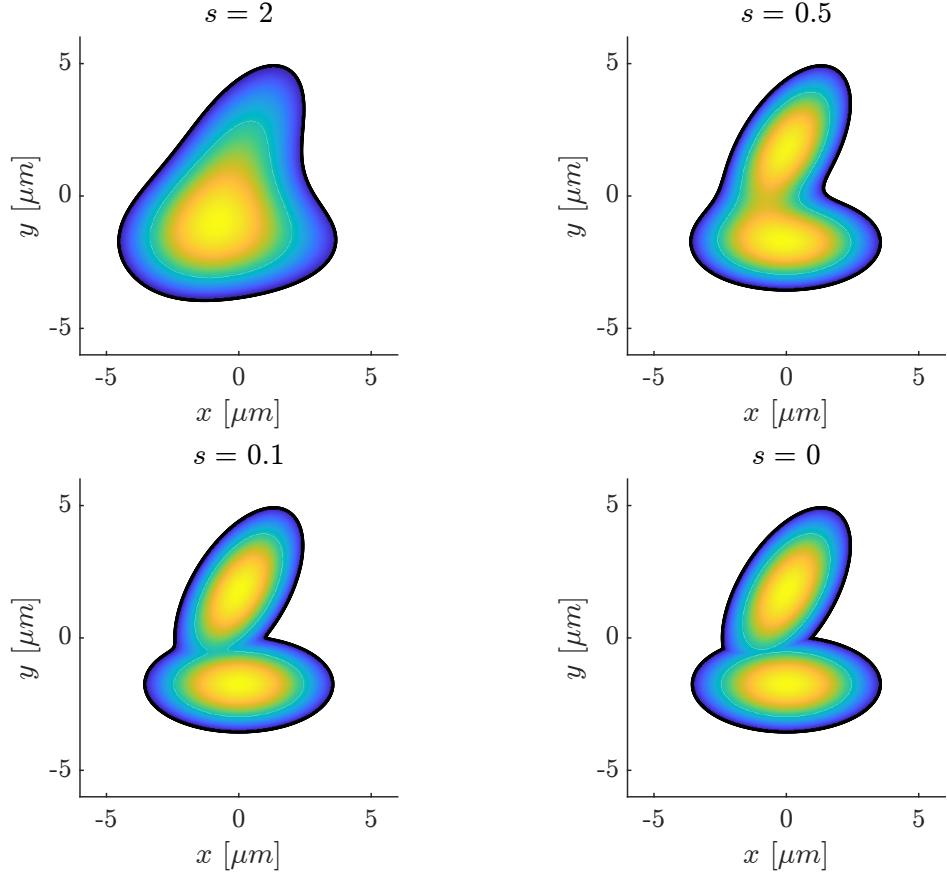


Figure 2.1: Two ellipses blended with various values of smoothness s . The bottom right figure is computed using standard minimum instead of smoothmin. Indeed, as $s \rightarrow 0$, the plot looks less smoothed across.

rotate the ellipse, using

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

where $\Delta\mathbf{x} = (x - x_c)\hat{\mathbf{i}} + (y - y_c)\hat{\mathbf{j}}$ and (x_c, y_c) is the center of the ellipse. We call the components of $\Delta\mathbf{x} = \Delta x\hat{\mathbf{i}} + \Delta y\hat{\mathbf{j}}$. The above transformation is a passive rotation because it rotates the whole SDF. The following equation

$$f(x, y) = \left[\frac{(x - x_c) \cos \theta + (y - y_c) \sin \theta}{p} \right]^2 + \left[\frac{-(x - x_c) \sin \theta + (y - y_c) \cos \theta}{q} \right]^2 - 1.$$

is used in the custom MATLAB function `ellipse.m` developed here. Each individual cell has a unique value of (x_c, y_c, p, q, θ) which we index by k . Bringing them all together

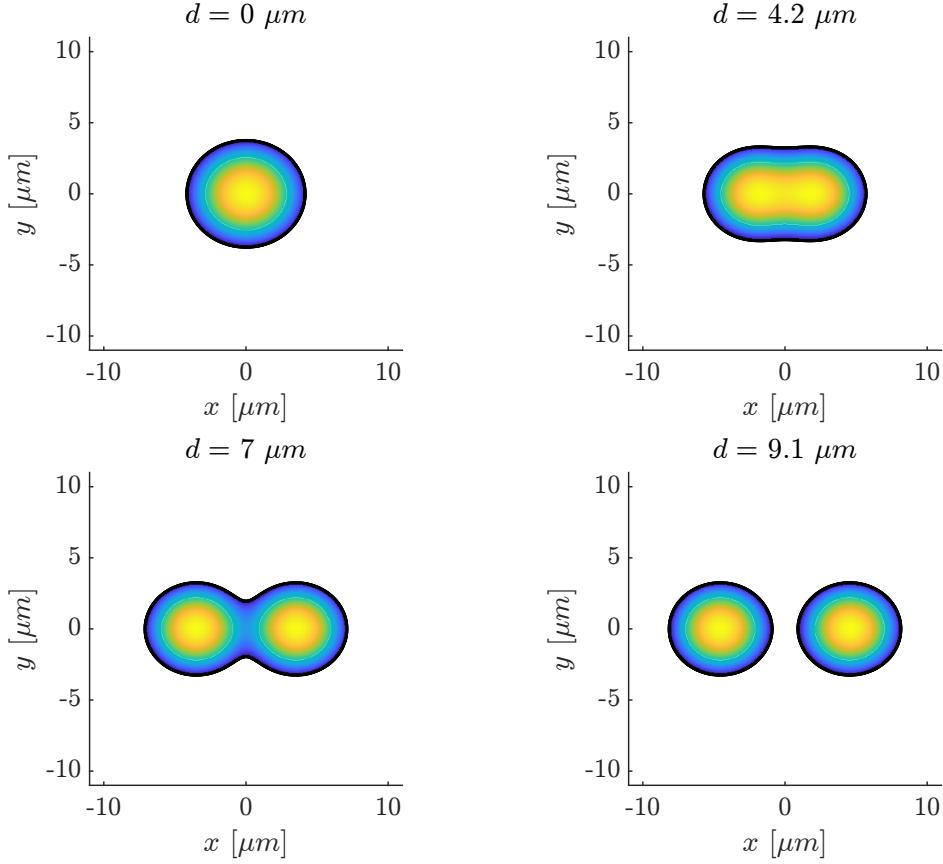


Figure 2.2: The separation d between two ellipse shapes with $s = 0.5$ and an aspect ratio of 0.9

in one function using [min](#), we have the colony SDF given by,

$$g(x, y) = \min_{k \in \{1, \dots, N_{\text{cells}}\}} f_k(x, y),$$

where the individual SDF for each cell is given by

$$\begin{aligned} f_k(x, y) &= \left[\frac{(x - x_{c,k}) \cos \theta_k + (y - y_{c,k}) \sin \theta_k}{p_k} \right]^2 \\ &+ \left[\frac{-(x - x_{c,k}) \sin \theta_k + (y - y_{c,k}) \cos \theta_k}{q_k} \right]^2 - 1. \end{aligned}$$

Cell colonies can also be built up by combining the SDFs of the individual cells using a cumulative smoothmin.

Some significant optimisations were made to ensure that the evaluation of the colony SDF (evaluated via calling `ellipse.m`) was as fast as possible within the capabilities of MATLAB. These optimisations, which were crucial because `ellipse.m` is called per time step, are commented on in Chapter 3, subsection 3.1.2.

2.2 Cell colony dynamics with an underlying discrete network

Now that we have introduced a robust mechanism to represent cell colony shape, the question naturally arises about how to represent the time dependence of this morphology. Baker's yeast grows via budding, as shown in figure 2.5. To represent the

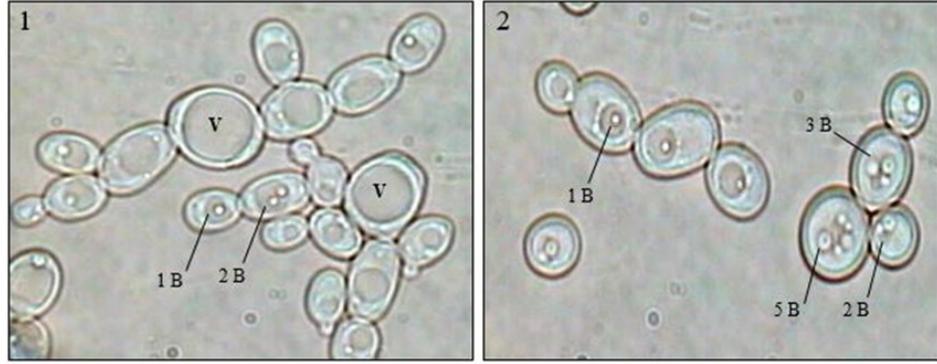


Figure 2.3: Micrographs of a translucent yeast strain studied in [14].

branching dynamics of pseudo-hyphal yeast growth, an underlying discrete network which can change node count has been developed. A colony for which the node count N_{nodes} remains fixed is given by an undirected graph with adjacency matrix A_{ij} where the edges are symbolically represented by springs which model the elasticity of the cells. A diagram of this is shown in figure 2.4.

Relating the positions of the nodes in the network is done based on a simple implementation shown in figure 2.5. The equations that represent this correspondence between a cell indexed k having parameters $(x_{c,k}, y_{c,k}, p_k, q_k, \theta_k)$ and its constitutive nodes indexed i_1 and i_2 are given by,

$$p_k(t) = \frac{1}{2} \|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|, \text{ where cell } k \text{ is comprised of nodes } i_1, i_2, \quad (2.1)$$

$$q_k(t) = \lambda_7 p_k(t), \quad (2.2)$$

$$\theta_k(t) = \text{atan}(y_{i_1} - y_{i_2}, x_{i_1} - x_{i_2}), \quad (2.3)$$

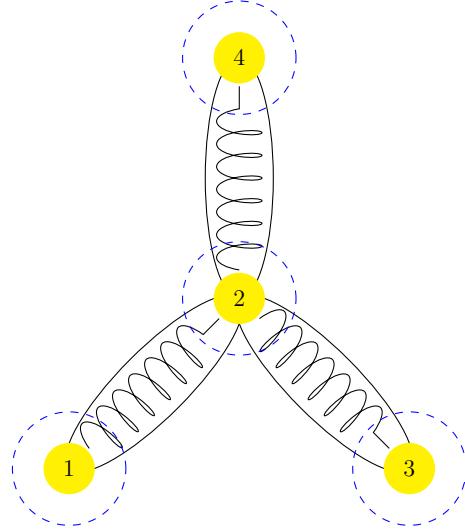


Figure 2.4: A diagram of three elliptical cells with internal springs to represent biomass elasticity ($\lambda_2 = \frac{K}{\eta\mu}$). The nodes are positioned at the ends of the major dimension of each ellipse and there are two nodes per cell. The force acting on node 2 for instance would be due to the forces from nodes 1, 3 and 4. The dashed blue circles represent the activation radius ($\lambda_4 = R/L_0$) of the contact force between the nodes.

$$\mathbf{x}_{c,k}(t) = \frac{1}{2} (\mathbf{x}_{i_1}(t) + \mathbf{x}_{i_2}(t)), \quad (2.4)$$

where atan is the 2 argument inverse tangent. In table 2.1 the variables associated with the cell indexed k , comprised of nodes i_1 and i_2 are summarised.

Symbol	Formula in terms of nodes i_1, i_2	Description
$\mathbf{x}_{c,k}$	$\frac{1}{2} (\mathbf{x}_{i_1} + \mathbf{x}_{i_2})$	Center
p_k	$\frac{1}{2} \ \mathbf{x}_{i_1} - \mathbf{x}_{i_2}\ $	Semi-major radius
q_k	$\frac{\lambda_7}{2} \ \mathbf{x}_{i_1} - \mathbf{x}_{i_2}\ $	Semi-minor radius
θ_k	$\text{atan}(y_{i_1} - y_{i_2}, x_{i_1} - x_{i_2})$	Orientation angle

Table 2.1: The variables associated with each cell indexed k and comprised of nodes i_1 and i_2 .

2.3 The colony adjacency matrix, A

In the software [CellColonySimulator](#) developed in this work, the maximum number of nodes which the simulation can take is set to `maxNodeCount = 500`. This number

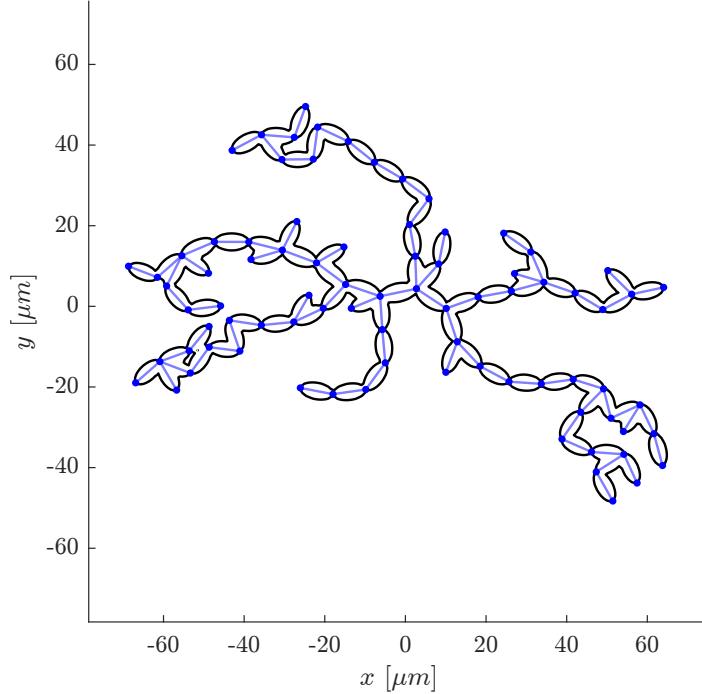


Figure 2.5: A plot showing the underlying network of a simulated yeast colony as well as the level-0 contour of the corresponding colony SDF.

is of course arbitrary and can be chosen to be much larger if supercomputing time is available. In the case of figure 2.4, suppose `maxNodeCount = 5`. Since there are only 4 active nodes, we would have an adjacency matrix of

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

which encodes the connectivity, i.e. node 1 is connected to node 2 and so on. In order to represent the spring force from nodes 1, 3 and 4 acting on node 2, we would sum up the spring forces from each connected node and apply Newton's Second Law, in the overdamped regime

$$\mathbf{v}_2 = \frac{\mathbf{F}_2}{\eta},$$

where \mathbf{v}_2 is the node 2's velocity, \mathbf{F}_2 is the net force acting on node 2, and η is the dampening constant which is assumed to be uniform in the simulation. The force on

node 2 due to node 1 for example is given by \mathbf{F}_{21} as

$$\mathbf{F}_{21} = -K (||\mathbf{x}_2 - \mathbf{x}_1|| - L_0) \frac{\mathbf{x}_2 - \mathbf{x}_1}{||\mathbf{x}_2 - \mathbf{x}_1||},$$

which is Hooke's Law for a spring of stiffness K and nominal length L_0 . The force acting on node 2 then is,

$$\mathbf{F}_2 = \mathbf{F}_{21} + \mathbf{F}_{23} + \mathbf{F}_{24},$$

where the forces \mathbf{F}_{23} and \mathbf{F}_{24} are similarly defined. For a system of nodes with arbitrary connectivity, the net force acting on node i is given by

$$\mathbf{F}_i = \sum_{j=1, j \neq i}^{N_{\text{nodes}}} A_{ij} \mathbf{F}_{ij},$$

where \mathbf{F}_{ij} is the spring force on node i due to node j .

In a very similar way, we add in a constant contact force between the nodes with radius $R \leq \frac{1}{2}L_0$, which ensures that the nodes do not overlap. This is sometimes called an exclusion principle. The third type of force is in a sense an external driving force for the colony. It is a force proportional to the nutrient concentration gradient $\nabla c(\mathbf{x}_i, t)$ at the cell position. This type of force represents the attraction of the cells to areas of the petri dish with a high nutrient concentration. A separate equation for the nutrient dynamics is introduced in section 2.5. All of the three types of forces: elastic, contact and chemotaxis are added together to determine the motion of each node.

$$\begin{aligned} \frac{d\mathbf{x}_i}{dt} = & \frac{K}{\eta} \sum_{j=1, j \neq i}^{N_{\text{nodes}}} \left[-A_{ij} (||\mathbf{x}_i - \mathbf{x}_j|| - L_0) \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||} \right] \\ & + \frac{F}{\eta} \sum_{j=1, j \neq i}^{N_{\text{nodes}}} \left[H(R - ||\mathbf{x}_i - \mathbf{x}_j||) \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||} \right] \\ & + \frac{\gamma}{\eta} \nabla c(\mathbf{x}_i, t), \end{aligned} \quad (2.5)$$

where F is the magnitude of the contact force, and γ is the magnitude of chemotaxis. In equation 2.5 we have used the kinematic relationship $\mathbf{v}_i = \frac{d\mathbf{x}_i}{dt}$ and H denotes a step function which is 0 for a negative argument and 1 for an argument in $\mathbb{R}_{\geq 0}$.

2.4 New cells from old

We add in nodes at a small distance $\delta = 0.01$ beside old nodes which are taken to be parent nodes. The addition of new nodes is done in global mitosis events at time

indices n that are a multiple of

$$\Delta n = \left\lceil \frac{1}{\Delta t} \right\rceil,$$

where Δt is the (non-dimensionalised) simulation time step, taken to be 0.01 in [CellColonySimulator](#). The number of cells added at each global mitosis event reflects the exponential growth equation,

$$N_{\text{nodes}} = N_{0,\text{cell}} e^{t\mu(t)},$$

where the time-dependent growth rate μ is discussed in more detail in Chapter 3, subsection 3.1.1. The growth rate itself is given to be proportional to the average nutrient concentration over the node indices, and is given by equation 2.13. This reflects the fact that if the nutrient has been depleted, no new cells should be added, whereas if the nutrient has a uniform concentration of $1.0 \mu\text{m}^{-2}$ (taken to be the ideal nutrient concentration for growth and explained in section 2.6), then the colony will grow at the ideal growth rate, given by μ^* .

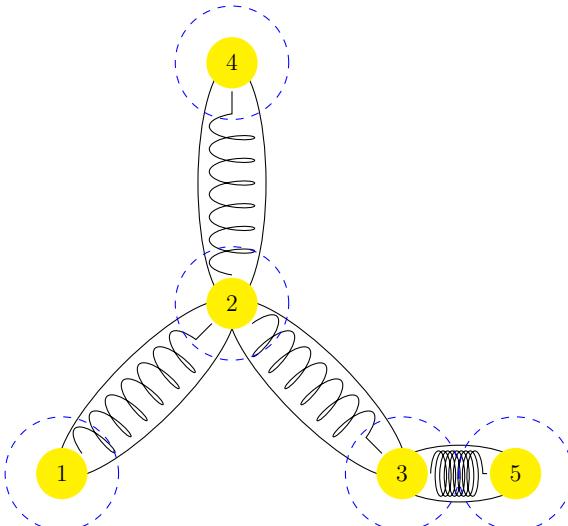


Figure 2.6: A mitosis event occurs via the addition of new nodes connected to old nodes. The nodes are added very close (distance $\delta \ll 1$) beside the original nodes (exaggerated here) so that the spring force can be defined. After this point, the initially compressed cell “grows” outwards to achieve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

Figure 2.6, demonstrates the mechanism by which a new node is added to the colony, in this case, node 5. The colony connectivity of this new 5-node network would

be given as

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

because node 5 is connected to node 3. We mention that the new colony is defined by a different equilibrium position (when the time-derivative of position is quasi-steady).

2.5 Incorporating the nutrient field

A nutrient medium containing glucose is assumed to be given by a reaction-diffusion partial differential equation. That is the nutrient concentration $c(x, y, t)$ is given by

$$\frac{\partial c(x, y, t)}{\partial t} = D \left(\frac{\partial^2 c(x, y, t)}{\partial x^2} + \frac{\partial^2 c(x, y, t)}{\partial y^2} \right) - rc(x, y, t)b(x, y, t),$$

where $b(x, y, t)$ 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 biomass field $b(x, y, t)$ is given as a function of the colony SDF as,

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

In the case of the plotting of $b(x, y, t)$ we use `nan` instead of 0.

The PDE is subjected to the Dirichlet condition that the nutrient density takes a value of 1.0 at the boundary. In order to simulate the nutrient field numerically a forward time centered space (FTCS) scheme was prototyped on the square grid covering the domain. However, this scheme was found to be numerically unstable for small spatial steps h and large time steps Δt and was discarded in favour of the unconditionally numerically stable Crank-Nicholson scheme detailed in section 2.7. Before this is possible, we carry out the non-dimensionalising of the equations of motion (EOMs).

2.6 Non-dimensionalising the equations of motion

In order to non-dimensionalise the equations of motion (EOMs) we introduce dimensionless parameters $\mathbf{x}_i = X\hat{\mathbf{x}}_i$, $t = T\hat{t}$, $b = B\hat{b}$ and $c = G\hat{c}$, where X, T, B and G are

general undetermined scalings for the independent and dependent variables, respectively. At the outset, we fix $X = L_0 = 7.5 \mu m$ the nominal major cell diameter found by [8]. The same source gives an aspect ratio of $7.5 \times 5.5 \mu m$ which we call $\lambda_7 = \frac{q}{p}$. The reason for the numbering will become clear from the non-dimensionalisation of the model. The time scale is chosen as $T = \frac{1}{\mu^*}$ the reciprocal of specific growth rate for yeast, that is the $\mu^* = \mu(0)$ that appears in the formula for the total node count. The value chosen was $\mu^* = 0.46 \text{ hour}^{-1}$ from the value cited at [41] which represents ideal conditions for *saccharomyces cerevisiae* growth on a synthetic culture medium containing glucose. The value given by [41] is a doubling rate of 90 mins. In the model presented in this thesis, that amounted to $\mu^* = \frac{\log(2)}{1.5 \text{ hours}} = 0.4621 \text{ hour}^{-1}$ which was rounded down to 0.46 hour^{-1} as a conservative estimate.

Symbol	Value	Descriptive name
L_0	$7.5 \mu m$	Average cell length
μ^*	0.46 hour^{-1}	Ideal growth rate
c_0	$1.0 \mu m^{-2}$	Nutrient concentration

Table 2.2: A summary of the numerical units

It is necessary to point out that initial concentration, the choice for $G = c_0$, is typically measured in $\text{mol} \cdot \mu m^{-2}$ which amounts to Avagadro's number of glucose molecules per μm^2 . In our case, we choose a number N_G such that the concentration comes out to $1.0 N_G \cdot \mu m^{-2}$. We always omit the N_G because it is of no consequence dimensionally. Also useful to recapitulate, is the formula for the total node count,

$$N_{\text{nodes}}(t) = N_{\text{cells},0} e^{t\mu^* \hat{\mu}(t)}, \quad (2.6)$$

where $N_{\text{cells},0} = 1$ because we always begin with a single cell, and $\hat{\mu}(t)$ is the unitless growth rate. Note that $\mu(t) = \mu^* \hat{\mu}(t)$ and that $\hat{\mu}(0) = 1.0$. This equation is used in the computation of μ^* based on the doubling time provided in [41]. Hopefully it is clear that the values of L_0, μ^*, c_0 can be changed to fit different species of fungus. Those values given here are just representative and further controlled experimental studies would need to be done to verify them.

2.6.1 Non-dimensionalising the nutrient PDE

Substituting these parameters into the reaction-diffusion PDE for nutrient concentration, we obtain

$$\frac{c_0}{(1/\mu^*)} \frac{\partial \hat{c}}{\partial \hat{t}} = \left(\frac{Dc_0}{L_0^2} \right) \left(\frac{\partial^2 \hat{c}}{\partial \hat{x}^2} + \frac{\partial^2 \hat{c}}{\partial \hat{y}^2} \right) - (rc_0B) \hat{c} \hat{b},$$

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 we 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

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

2.6.2 Non-dimensionalising the nodes ODE

Now, for the EOM for the nodes, we have

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

Substituting the dimensionless parameters, we acquire

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

which, after rearranging, becomes,

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

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,

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

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

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

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

2.6.3 Non-dimensionalising the biomass equation

The field $g(x, y, t) = \hat{g}(\hat{x}, \hat{y}, \hat{t})$ is unitless so we are left with the following EOM for the biomass field,

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

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

which leaves us with an additional parameter, $\lambda_6 = \frac{r}{\mu^*}$. The final model parameter is the cell aspect ratio, λ_7 . The model parameters are summarised in table 2.3.

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	$\frac{R}{L_0}$	repulsion radius
λ_5	$\frac{\gamma c_0}{\eta\mu^*L_0^2}$	mobility
λ_6	$\frac{r}{\mu^*}$	metabolic rate
λ_7	$\frac{\text{cell minor axis}}{\text{cell major axis}} = \frac{2q}{2p}$	cell aspect ratio

Table 2.3: A summary of the dimensionless model parameters

2.6.4 Non-dimensional model equations of motion (EOMs)

Dropping the hats on variable symbols, we arrive at the final system equations of motion which span from equation 2.7 to 2.20.

$$\frac{\partial c}{\partial t} = \lambda_1 \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) - bc, \text{ for } (x, y) \in \Omega \setminus \partial\Omega, \quad (2.7)$$

$$c(x, y, t) = 1.0, \text{ for } (x, y) \in \partial\Omega, \quad (2.8)$$

$$c(x, y, 0) = 1.0, \quad (2.9)$$

$$\begin{aligned} \frac{d\mathbf{x}_i}{dt} &= \lambda_2 \sum_{j=1, j \neq i}^{N_{\text{nodes}}} \left[A_{ij} (1 - \|\mathbf{x}_i - \mathbf{x}_j\|) \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \right] \\ &+ \lambda_3 \sum_{j=1, j \neq i}^{N_{\text{nodes}}} \left[H(\lambda_4 - \|\mathbf{x}_i - \mathbf{x}_j\|) \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \right] \\ &+ \lambda_5 \nabla c(\mathbf{x}_i, t), \text{ for } i \in \{1, \dots, N_{\text{nodes}}\}, \end{aligned} \quad (2.10)$$

$$\mathbf{x}_1(0) = (0.5 \cos \Theta_e, 0.5 \sin \Theta_e), \mathbf{x}_2(0) = (-0.5 \cos \Theta_e, -0.5 \sin \Theta_e), \quad (2.11)$$

$$N_{\text{nodes}} = N_{0, \text{cells}} e^{t\mu(t)}, \text{ where } N_{0, \text{cells}} = 1, \quad (2.12)$$

$$\mu(t) = \frac{1}{N_{\text{nodes}}} \sum_{i=1}^{N_{\text{nodes}}} c(\mathbf{x}_i, t), \quad (2.13)$$

$$b(x, y, t) = \begin{cases} -\lambda_6 g(x, y, t), & \text{if } g(x, y, t) \leq 0, \\ 0, & \text{otherwise,} \end{cases} \quad (2.14)$$

$$g(x, y, t) = \min_{k \in \{1, \dots, N_{\text{cells}}\}} f_k(x, y, t), \quad (2.15)$$

$$\begin{aligned} f_k(x, y, t) &= \left[\frac{(x - x_{c,k}(t)) \cos \theta_k + (y - y_{c,k}(t)) \sin \theta_k}{p_k(t)} \right]^2 \\ &+ \left[\frac{-(x - x_{c,k}(t)) \sin \theta_k + (y - y_{c,k}(t)) \cos \theta_k}{q_k(t)} \right]^2 - 1, \end{aligned} \quad (2.16)$$

$$p_k(t) = \frac{1}{2} \|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|, \text{ where cell } k \text{ is comprised of nodes } i_1, i_2, \quad (2.17)$$

$$q_k(t) = \lambda_7 p_k(t), \quad (2.18)$$

$$\theta_k(t) = \text{atan}(y_{i_1} - y_{i_2}, x_{i_1} - x_{i_2}), \quad (2.19)$$

$$\mathbf{x}_{c,k}(t) = \frac{1}{2} (\mathbf{x}_{i_1}(t) + \mathbf{x}_{i_2}(t)). \quad (2.20)$$

Note that i and j index over node numbers, k over cells, and i_1 and i_2 denote the nodes that comprise cell k . Also note that Θ_e , the orientation of the starting cell is chosen to be different for each ensemble instance e . In fact, this is chosen to be a uniform random number between $[0, 2\pi]$ which is updated in the loop over the ensemble in `Master.m` as part of the **CellColonySimulator** MATLAB software. Finally the petri-dish domain is given by

$$\Omega = \left[-\frac{1}{2}L_{\text{petri-dish}}, \frac{1}{2}L_{\text{petri-dish}} \right] \times \left[-\frac{1}{2}L_{\text{petri-dish}}, \frac{1}{2}L_{\text{petri-dish}} \right] \subset \mathbb{R}^2, \quad (2.21)$$

where it is noted that $L_{\text{petri-dish}}$ is unitless.

2.7 Numerically solving for the nutrient field

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 $c(x, y, t)$ 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 [11] 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-Nicolson scheme is unconditionally numerically stable. Let us begin by verifying this fact using what is known as Von Neumann stability analysis [7].

Firstly, we define the nutrient field as $c_{l,m}^n = c(x_l, y_m, t_n)$, where $x_l = -\frac{1}{2}L_{\text{petri-dish}} + hl$, $y_m = -\frac{1}{2}L_{\text{petri-dish}} + hm$ and similarly for $b_{l,m}^n$, the biomass. The values $c_{l,m}^n$ precisely satisfy the discretised equations which follow. Let $\bar{c}_{l,m}^n$ be the solution achieved with finite precision arithmetic operations which does not necessarily satisfy the equations, up to some error term,

$$\epsilon_{l,m}^n = \bar{c}_{l,m}^n - c_{l,m}^n.$$

Carrying out the Crank-Nicolson discretisation with the values $c_{l,m}^n$, we derive

$$\frac{c_{l,m}^{n+1} - c_{l,m}^n}{\Delta t} = \frac{\lambda_1}{2} (\nabla^2 c_{l,m}^{n+1} + \nabla^2 c_{l,m}^n) - \frac{1}{2} b_{l,m}^n (c_{l,m}^{n+1} + c_{l,m}^n).$$

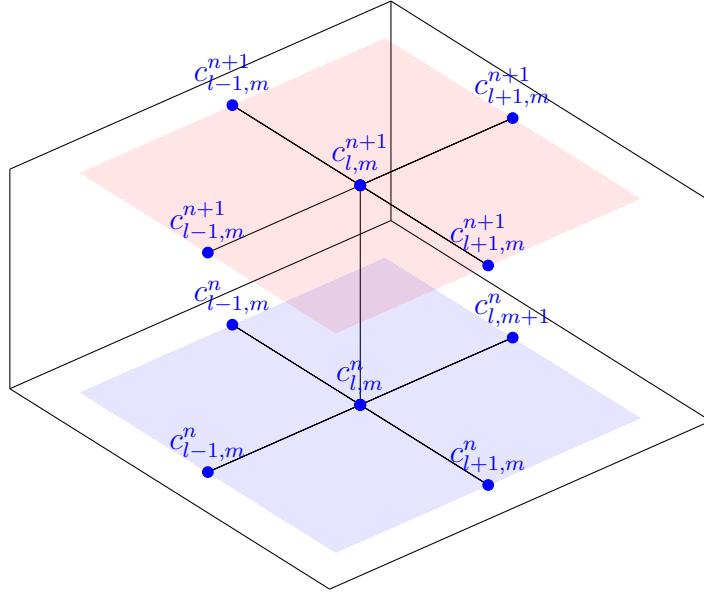


Figure 2.7: The five point stencil for the nutrient field at times $t_n = n\Delta t$ (blue plane) and $t_{n+1} = (n + 1)\Delta t$ (red plane) used for the Crank-Nicolson scheme on the domain interior.

The Laplacian terms here are calculated with a five-point stencil at the current and subsequent time steps as shown in figure 2.7. This results in the following discretisation,

$$\begin{aligned} \frac{c_{l,m}^{n+1} - c_{l,m}^n}{\Delta t} &= \frac{\lambda_1}{2} \left(\frac{c_{l+1,m}^{n+1} - 2c_{l,m}^{n+1} + c_{l-1,m}^{n+1}}{h^2} + \frac{c_{l,m+1}^{n+1} - 2c_{l,m}^{n+1} + c_{l,m-1}^{n+1}}{h^2} \right) \\ &\quad + \frac{\lambda_1}{2} \left(\frac{c_{l+1,m}^n - 2c_{l,m}^n + c_{l-1,m}^n}{h^2} + \frac{c_{l,m+1}^n - 2c_{l,m}^n + c_{l,m-1}^n}{h^2} \right) \\ &\quad - \frac{1}{2} b_{l,m}^n (c_{l,m}^{n+1} + c_{l,m}^n). \end{aligned}$$

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

$$\begin{aligned} c_{l,m}^{n+1} - c_{l,m}^n &= \frac{\alpha}{2} (c_{l+1,m}^{n+1} + c_{l-1,m}^{n+1} + c_{l,m+1}^{n+1} + c_{l,m-1}^{n+1}) - 2\alpha c_{l,m}^{n+1} \\ &\quad + \frac{\alpha}{2} (c_{l+1,m}^n + c_{l-1,m}^n + c_{l,m+1}^n + c_{l,m-1}^n) - 2\alpha c_{l,m}^n \\ &\quad - \frac{\Delta t}{2} b_{l,m}^n (c_{l,m}^{n+1} + c_{l,m}^n). \end{aligned}$$

Since the discretised equations are linear in $c_{l,m}^n$ (taking $b_{l,m}^n$ to be slowly varying) and

since $\bar{c}_{l,m}^n$ also form a solution, the $\epsilon_{l,m}^n$ are also a solution.

$$\begin{aligned}\epsilon_{l,m}^{n+1} - \epsilon_{l,m}^n &= \frac{\alpha}{2} (\epsilon_{l+1,m}^{n+1} + \epsilon_{l-1,m}^{n+1} + \epsilon_{l,m+1}^{n+1} + \epsilon_{l,m-1}^{n+1}) - 2\alpha\epsilon_{l,m}^{n+1} \\ &\quad + \frac{\alpha}{2} (\epsilon_{l+1,m}^n + \epsilon_{l-1,m}^n + \epsilon_{l,m+1}^n + \epsilon_{l,m-1}^n) - 2\alpha\epsilon_{l,m}^n \\ &\quad - \frac{\Delta t}{2} b_{l,m}^n (\epsilon_{l,m}^{n+1} + \epsilon_{l,m}^n).\end{aligned}$$

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

$$\epsilon_{l,m}^n = \xi^n e^{i\nu_x l h} e^{i\nu_y m h},$$

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,

$$\begin{aligned}\xi^{n+1} e^{i\nu_x l h} e^{i\nu_y m h} - \xi^n e^{i\nu_x l h} e^{i\nu_y m h} &= \frac{\alpha}{2} (\xi^{n+1} e^{i\nu_x(l+1)h} e^{i\nu_y m h} + \xi^{n+1} e^{i\nu_x(l-1)h} e^{i\nu_y m h}) \\ &\quad + \frac{\alpha}{2} (\xi^{n+1} e^{i\nu_x l h} e^{i\nu_y(m+1)h} + \xi^{n+1} e^{i\nu_x l h} e^{i\nu_y(m-1)h}) \\ &\quad - 2\alpha\xi^{n+1} e^{i\nu_x l h} e^{i\nu_y m h} \\ &\quad + \frac{\alpha}{2} (\xi^n e^{i\nu_x(l+1)h} e^{i\nu_y k h} + \xi^n e^{i\nu_x(m-1)h} e^{i\nu_y m h}) \\ &\quad + \frac{\alpha}{2} (\xi^n e^{i\nu_x l h} e^{i\nu_y(k+1)h} + \xi^n e^{i\nu_x m h} e^{i\nu_y(m-1)h}) \\ &\quad - 2\alpha\xi^n e^{i\nu_x l h} e^{i\nu_y k h} \\ &\quad - \frac{\Delta t}{2} b_{l,m}^n (\xi^{n+1} e^{i\nu_x l h} e^{i\nu_y m h} + \xi^n e^{i\nu_x l h} e^{i\nu_y m h}).\end{aligned}$$

Dividing through by $\xi^n e^{i\nu_x l h} e^{i\nu_y m h}$, we attain,

$$\begin{aligned}\xi - 1 &= \frac{\alpha}{2} (\xi e^{i\nu_x h} + \xi e^{-i\nu_x h}) + \frac{\alpha}{2} (\xi e^{i\nu_y h} + \xi e^{-i\nu_y h}) - 2\xi\alpha \\ &\quad + \frac{\alpha}{2} (e^{i\nu_x h} + e^{-i\nu_x h}) + \frac{\alpha}{2} (e^{i\nu_y h} + e^{-i\nu_y h}) - 2\alpha - \frac{\Delta t}{2} b_{l,m}^n (\xi + 1).\end{aligned}$$

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,

$$\begin{aligned}\xi - 1 &= \alpha\xi \cos(\nu_x h) + \alpha\xi \cos(\nu_y h) - 2\alpha\xi \\ &\quad + \alpha \cos(\nu_x h) + \alpha \cos(\nu_y h) - 2\alpha - \frac{\Delta t}{2} b_{l,m}^n (\xi + 1).\end{aligned}$$

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

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

In order to complete the analysis, consider when biomass is $b_{l,m}^n = 0$. 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 $b_{l,m}^n$ 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.

Chapter 3

Software Implementation: *CellColonySimulator*

3.1 The structure of the program

The program **CellColonySimulator** is implemented *in-house* with MATLAB as shown in figure 3.1. **CellColonySimulator** is broken up into six separate scripts as per the programming principle of modularity. These include: `Master.m`, `runSimulation.m`, `ellipse.m`, `updateFood.m`, `addCell.m`, `updateNodes.m`.

3.1.1 Calling the update loop in *runSimulation.m*

The function `runSimulation.m` takes in the model parameter vector, λ , the struct of numerical discretisation parameters, `numericPack`, the times at which statistics are sampled, `sampleTimes`, the colony data including node positions (x, y) and the network adjacency matrix, `connectivity`, and the initial nutrient field given by `food`.

The mesh grid matrices X and Y are taken from `numericPack` and converted to graphics processing unit (GPU) arrays using MATLAB's `gpuArray()`. This is crucial for speed as GPU operations on large matrices are highly optimised and parallelised within the GPU architecture. Besides generating descriptive file names for the output plots and other miscellaneous tasks, `runSimulation.m` then progresses into the primary `for` loop over the time indices, `timeIndex`.

A task to print to the console to let the user know what ensemble instance and `timeIndex` the program is up to, is done first. Secondly, based on the current node positions (x_i, y_i) for $i \in \{1, \dots, N_{\text{nodes}}\}$, and the number of edges present in the `connectivity` matrix, the current number of cells N_{cells} is counted up and vectors are associated to

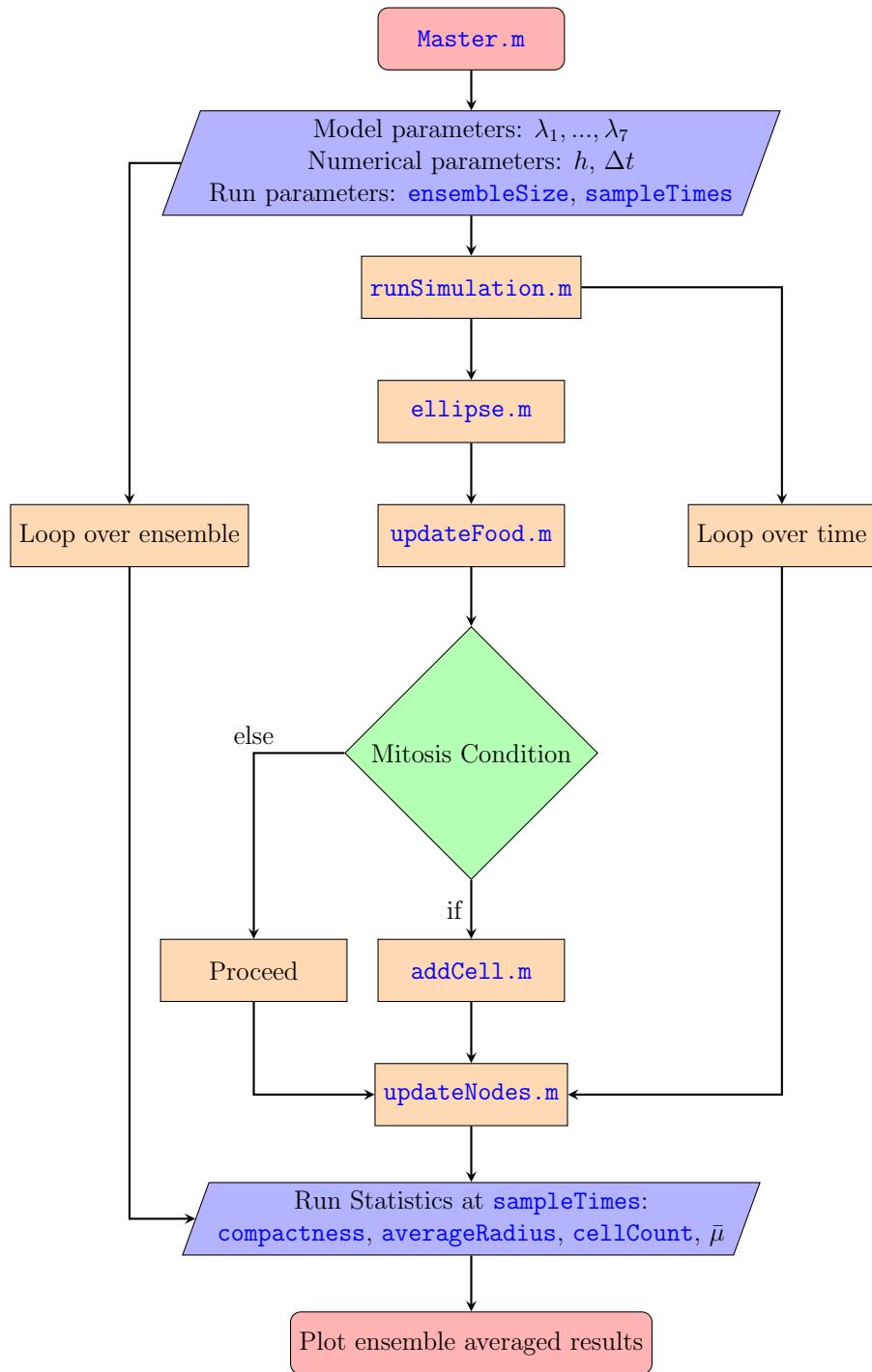


Figure 3.1: A flow chart of **CellColonySimulator** software package in MATLAB.

the variables $x_{c,k}, y_{c,k}$, θ_k , p_k and q_k (based on equations 2.17, 2.18, 2.19 and 2.20) where $k \in \{1, \dots, N_{\text{cells}}\}$. These vectors are passed to the function `ellipse()` (explained in subsection 3.1.2) which computes the colony-level signed distance field (SDF), which is comprised of the elliptical SDF of each cell indexed k .

At this stage the biomass field, `biomass`, is computed from the colony level SDF, based on equation 2.14. The current value of the nutrient field and the biomass field, are passed to `updateFood()` (explained in subsection 3.1.3) which updates the nutrient field based on the Crank-Nicolson method applied to equation 2.7. The food at the next time step is the output from `updateFood()`.

The value of the growth rate across the colony μ_i is then determined using 2D interpolation by calling `interp2(X, Y, food, x, y)`, where X, Y are the underlying grid matrices, `food` is the updated nutrient field, and (x, y) are the node positions, taken as query points in the interpolation. Recall from equation 2.13, the growth rate is given by the the average of the nutrient field sampled over the nodes.

The updated number of nodes, $N'_{\text{nodes}} = N_{\text{nodes}}(t_{n+\Delta n})$, is found by multiplying the current number of nodes, N_{nodes} , by e^μ (as stipulated by the exponential growth in equation 2.12) where μ is the average growth rate over the colony as discussed above. The number of nodes added is then given by

$$\Delta N_{\text{nodes}} = N'_{\text{nodes}} - N_{\text{nodes}} = \lceil e^\mu - 1 \rceil N_{\text{nodes}},$$

where μ is given by

$$\mu = \frac{1}{N_{\text{nodes}}} \sum_{i=1}^{N_{\text{nodes}}} \mu_i,$$

where $\mu_i = c(\mathbf{x}_i, t)$ is the value of the nutrient field $c(\mathbf{x}_i, t)$ sampled at the i -th node position (this is just a recapitulation of equation 2.13). The number of time steps per mitosis event, Δn is given by `ceil(1.0/dt)`, and we use an `if` statement to ensure `addCell()` is only called when n is a multiple of Δn .

In `addCell()`, the number of nodes to add, ΔN_{nodes} , is passed as a argument. The mechanism to choose the parent cells to undergo mitosis is explained in subsection 3.1.4. Finally, the new node positions (based on equation 2.10) are determined using `updateNodes()` as detailed in subsection 3.1.5.

The rest of `runSimulation.m` is devoted to plotting the biomass and nutrient fields, as well as generating the underlying node network using MATLAB's `graph()` where the colony adjacency matrix, `connectivity`, is passed as an argument.

3.1.2 Vectorised computation of colony SDF in *ellipse.m*

The custom function `ellipse.m` has been iterated with high efficiency in mind since `ellipse.m` has to be called every time step. Using MATLAB's widget *run-and-time*, the bottleneck found in `ellipse.m` was eliminated. This significantly improved run times. How was this speedup achieved?

Where previously `ellipse.m` required a `for` loop over the SDFs for each cell, the final iteration of `ellipse.m` uses completely vectorised operations which run fast on the GPU. The code is provided in listing 3.1.

Listing 3.1: A code listing for the `ellipse.m` script

```

1  function sdf =                      ... %Colony level SDF
2    ellipse(X, Y,                      ... %Grid values
3        theta,                         ... %Orientation angles
4        centerX, centerY, ... %Center coordinates
5        semiMajorRadius, ... %p_k
6        semiMinorRadius ... %q_k
7    )
8    %Translate the field to the center coordinates:
9    Xt = repmat(X, [1,1,size(centerX,2)])-reshape(centerX,
10      [1,1,size(centerX,2)]);
11    Yt = repmat(Y, [1,1,size(centerX,2)])-reshape(centerY,
12      [1,1,size(centerX,2)]);
13    %Precompute trigonometric functions for efficiency
14    COS = cos(theta);
15    SIN = sin(theta);
16    cosReshape = reshape(COS, [1,1,size(centerY,2)]);
17    sinReshape = reshape(SIN, [1,1,size(centerY,2)]);
18    %Rotate translated field by the rotation matrix (
19      correctly vectorised):
20    rotX = Xt .* cosReshape + Yt .* sinReshape;
21    rotY = -Xt .* sinReshape + Yt .* cosReshape;
22    %Precompute reciprocal squared of geometry arrays:
23    semiMajorRadiusReciprocalReshape2 = ...
24      reshape(1.0 ./ semiMajorRadius.^2, [1,1,size(
          centerX,2)]);
25    semiMinorRadiusReciprocalReshape2 = ...
26      reshape(1.0 ./ semiMinorRadius.^2, [1,1,size(
          centerX,2)]);
27    %Instead of using smoothmin (slow) just use min of the

```

```

25 %square of the SDF. [This does not require calling sqrt
26   ()
27 %which is computationally very costly]
28 sdf = min(rotX.^2 .* semiMajorRadiusReciprocalReshape2
29   + ...
30     rotY.^2 .* semiMinorRadiusReciprocalReshape2
31   - ...
32   1.0, [], 3);
33 %Normalise to with magnitude 1 in colony region:
34 absMin = abs(min(sdf,[], "all"));
35 sdf = sdf ./ absMin ;
36 end

```

One thing to note from the MATLAB code listing is that vectorised division, say called with `./`, is precomputed as reciprocals which are then multiplied through in the computation of `sdf` using `.*` which is faster. Precomputing trigonometric functions was also opted for with the aim of reducing repeated computations with the same angle. Finally, we recall the discussion in section 2.1, in which it was noted that `min` is significantly faster than evaluating `smoothmin`. The final optimisation made was to remove the call to `sqrt()` which actually has no effect on the shape of each cell but vastly improves compute time.

3.1.3 A fast Crank-Nicolson scheme in `updateFood.m`

The Crank-Nicolson scheme is an implicit numerical scheme that requires the computation of a (mostly zero) matrix of coefficients `LHM` (which has dimensions of about $N_{\text{grid}}^2 \times N_{\text{grid}}^2$) a right hand side column vector `RHS` ($N_{\text{grid}}^2 \times 1$), and the solution of the linear system `LHM * x = RHS`.

The most obvious optimisation made in `updateFood()` is to completely vectorise all operations. As evinced in the code given in listing 3.2, there are no `for` loops.

Listing 3.2: A code listing for the `updateFood.m` script. The variable name `CFL` refers to the Courant–Friedrichs–Lewy number which comes up in numerical schemes. This is called α in Chapter 2.

```

1 function foodNew = updateFood(biomass, food, lambda, dt, h, X)
2   N = size(X,1);
3   V = N .* N;
4   Vi = (N-2) .* (N-2);
5   i = 2:(N-1);
6   j = i;
7   %Generate the five point stencil of indices in row major order.

```

```

8    ij    = rmo(i,j,N);
9    im1j = rmo(i-1,j,N);
10   ip1j = rmo(i+1,j,N);
11   ijm1 = rmo(i,j-1,N);
12   ijp1 = rmo(i,j+1,N);
13 %Flatten these:
14   fij   = ij(:);
15   fim1j = im1j(:);
16   fip1j = ip1j(:);
17   fijm1 = ijm1(:);
18   fijp1 = ijp1(:);
19
20 CFL = dt .* lambda(1) ./ (h .* h);
21 coeff1 = 0.5 .* dt .* biomass(i,j) + 2.0 .* CFL + 1.0;
22 coeff2 = 0.5 .* CFL;
23 coeff3 = 1.0 - 2.0 .* CFL - 0.5 .* dt .* biomass(i,j);
24 %Flatten the coefficients:
25 coeff1 = coeff1(:);
26 coeff3 = coeff3(:);
27 c = food(:);
28
29 RHS = coeff3 .* c(fij) + ...
30       coeff2 .* (c(fim1j) + c(fip1j) + c(fijm1) + c(fijp1));
31
32 LHM = spdiags([-coeff2 .* ones(Vi,1) ,-coeff2 .* ones(Vi,1) ,...
33                  gather(coeff1),...
34                  -coeff2 .* ones(Vi,1) ,-coeff2 .* ones(Vi,1) ],...
35                  [-N, -1, 0, 1, N], Vi, Vi);
36 yM = spdiags(-coeff2 .* ones(1,N), 0, N, N);
37 LHM(1:N,(end - N + 1):end) = LHM(1:N,(end - N + 1):end) + yM;
38 LHM((end - N + 1):end,1:N) = LHM((end - N + 1):end,1:N) + yM;
39
40 [cNew, flag] = pcg(LHM,RHS,1e-6);
41
42 foodNew = food;
43
44 foodNew(i,j) = reshape(cNew, [N-2, N-2]);
45 end
46
47
48 function index = rmo(i,j,N)
49     index = i' + N .* (j-1);
50 end

```

Another optimisation which was not only efficient but also necessary to avoid memory issues, was to construct `LHM` with sparse matrices, which meant that on the order of N_{grid}^4 redundant zeros did not have to be stored in memory. In particular, the pentadiagonal matrix (5 non-zero diagonals) `LHM` was constructed using `spdiags`. A crucial optimisation from the point of view of performance was to replace the direct linear solution, `LMH \ RHS`, with the function, `pcg`, which carries out the preconditioned conjugate gradient numerical method at no appreciable change in accuracy (error bounded by `1e-6`), but significantly faster.

Regarding indexing, note that MATLAB uses column-major-order by default in its vectorised functions (see MathWorks website). To align with this convention, the row-major-order function, `rmo`, transposes the row indices, `i`, and leaves the column indices, `j`, fixed. Other configurations were trialled but this was ultimately the only one that produced the correct nutrient diffusion dynamics.

3.1.4 Introducing new nodes in `addCell.m`

Adding in ΔN_{nodes} new nodes to the colony requires the selection of ΔN_{nodes} parent nodes to undergo mitosis. This choice is based on first interpolating the nutrient available to each node, and using `maxk()` to decide on the ΔN_{nodes} parent node indices corresponding to greatest values of `food`. These are called `buddingIndices` in the `addCell.m` script. The new node positions are defined by first generating a random budding angle, θ_b , for each node to be added. The formula for a child node's position is given by

$$\begin{aligned}x_{\text{child}} &= x_{\text{parent}} + \delta \cos \theta_b, \\y_{\text{child}} &= y_{\text{parent}} + \delta \sin \theta_b,\end{aligned}$$

where $\delta = 0.01$ is a very small dislocation radius.

Finally, the addition of new nodes implies the addition of new cells which are defined by the edge connecting a pair of nodes. This change in the adjacency matrix requires updating the `connectivity` array by placing a 1 in the correct positions. Recall that the `connectivity` array is a symmetric logical matrix.

3.1.5 Advancing the node positions using Euler's method in `updateNodes.m`

The `updateNodes.m` script employs Euler's first order time marching algorithm to update the node positions, based on three forces: elasticity, contact and chemotaxis. It was conceivable that a Runge-Kutta-4 (RK4) explicit time stepping scheme, detailed in [51], could have been implemented, however this was deemed to be unnecessary since

determining the overdamped spring dynamics to a high degree of accuracy is always swamped by the overall conceptual errors in the modelling. From a mathematical modelling point of view, the spring has been chosen to ensure the passage of the colony from one equilibrium state at the old number of nodes, N_{nodes} , to the equilibrium state at the updated node count, N'_{nodes} . The Euler method was sufficient to allow this transition without any numerical stability issues. The code is given in listing 3.3.

Listing 3.3: A code listing for the **updateNodes.m** script

```

1 function [xNew,yNew] = updateNodes(x, y, lambda, connectivity, food
2 , numNodes, dt, h, X,Y)
3 %Firstly calculate the matrix of distances between the nodes;
4 xNodes = x(1:numNodes);
5 yNodes = y(1:numNodes);
6 xMatrix = gpuArray(xNodes - xNodes');
7 yMatrix = gpuArray(yNodes - yNodes');
8 matrixOfSquares = xMatrix.^2 + yMatrix.^2;
9 matrixOfSquaresDivisor = matrixOfSquares + (matrixOfSquares ==
10 0);
11 rSQRT = 1 ./ sqrt(matrixOfSquaresDivisor);
12
13 xNew = x;
14 yNew = y;
15 contactMask = double(matrixOfSquares <= lambda(4).^2);
16 connectivityNodes = double(connectivity(1:numNodes,1:numNodes))
17 );
18
19 %Add the force due to the spring:
20 springTermX = (1.0 - rSQRT) .* xMatrix;
21 springTermX = springTermX .* connectivityNodes;
22 springTermY = (1.0 - rSQRT) .* yMatrix;
23 springTermY = springTermY .* connectivityNodes;
24
25 xNew(1:numNodes) = xNew(1:numNodes) - ...
26 dt .* lambda(2) .* sum(springTermX, 2);
27 yNew(1:numNodes) = yNew(1:numNodes) - ...
28 dt .* lambda(3) .* sum(springTermY, 2);
29
30 contactTermX = rSQRT .* xMatrix;
31 contactTermX = contactTermX .* contactMask;
32 contactTermY = rSQRT .* yMatrix;
33 contactTermY = contactTermY .* contactMask;
34
35 %Add in the contact force:
```

```

33 xNew(1:numNodes) = xNew(1:numNodes) + ...
34     dt .* lambda(1) .* sum(contactTermX, 2);
35 yNew(1:numNodes) = yNew(1:numNodes) + ...
36     dt .* lambda(1) .* sum(contactTermY, 2);
37
38 [foodGradientX, foodGradientY] = gradient(food, h);
39
40 %Sample the gradient at the nodes:
41 foodGradientXAtNodes = interp2(X,Y,foodGradientX, xNodes,
42     yNodes );
42 foodGradientYAtNodes = interp2(X,Y,foodGradientY, xNodes,
43     yNodes );
44
45 %Add in the chemotactic force:
46 xNew(1:numNodes) = xNew(1:numNodes) + dt .* lambda(5) .* 
47     foodGradientXAtNodes;
46 yNew(1:numNodes) = yNew(1:numNodes) + dt .* lambda(5) .* 
47     foodGradientYAtNodes;
47 end

```

Note that the `matrixOfSquaresDivisor` is constructed in such a way as to avoid introducing `nan` values into the solution.

3.2 Statistics from the model

The software developed thus far is a numerical framework 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 separate run of the model starting with identical initial conditions will look different after some time. It is expected, however, that some “averaged” quantity will stabilise so long as the model is simulated over a fairly large number of runs. Separate runs of the model belong to a set called an ensemble.

A fully grown colony will in general not be perfectly circular in shape. In order to measure the roundness of the colony (or analogously, the degree of branching) we use the compactness metric used in image processing (for example, see [28]). This is given by

$$C = \frac{P^2}{4\pi A}, \quad (3.1)$$

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 calcu-

lated from based on where the colony SDF changes sign.

A black and white image is produced at each time step using Matlab's function `bwboundaries` as per Li's implementation, [28]. The area then is given by summing up the grid squares that are inside the implicit shape given by the SDF and multiplying by h^2 . The perimeter is obtained 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 3.2 demonstrates visually that more branched shapes have higher compactness than approximately circular shapes. The

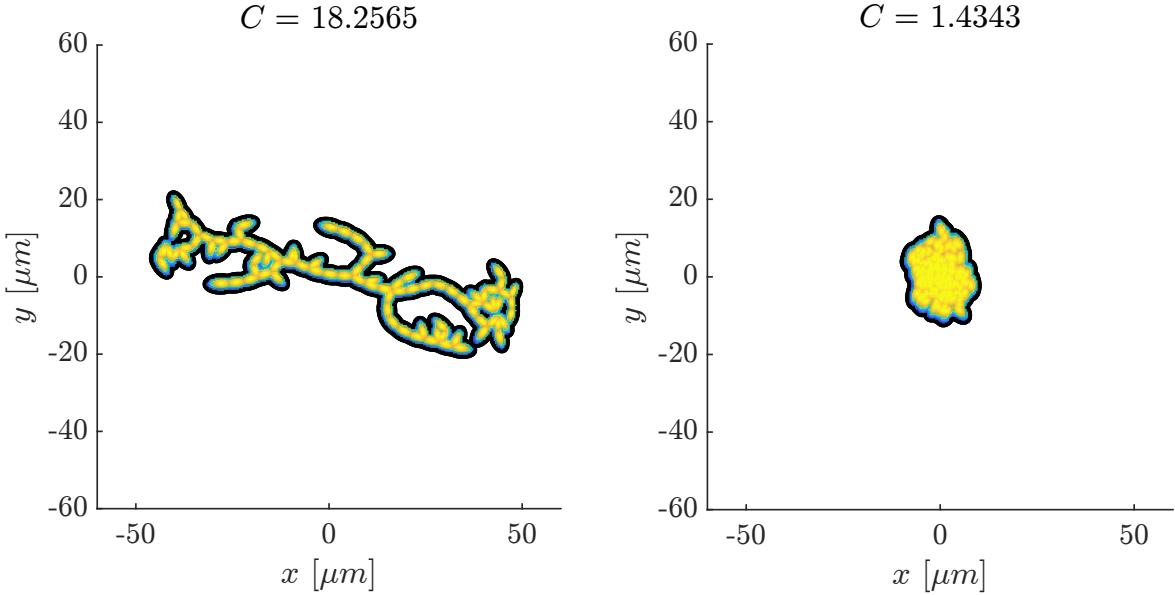


Figure 3.2: A comparsion of high and low compactness

summary statistics descriptions are collected in table 3.1.

Symbol	Description
C	Colony compactness.
μ	Colony average growth rate.
\bar{d}	Average distance of cells from origin.
N_{cells}	Cell count.

Table 3.1: The summary statistics extracted from the model

Chapter 4

Numerical experiments and simulation results

In this section the outputs of `CellColonySimulator` are presented. The figures 4.1, 4.2, 4.3, 4.4 and 4.5 show the growth over time of the biomass field $b(x, y, t)$ and the nutrient field $c(x, y, t)$ for various values of $\lambda_1, \dots, \lambda_7$. As stated in Chapter 1, a nominal length of $L_0 = 7.5 \mu m$ is used for all runs. Furthermore, the (unitless) time step and spatial step used are $\Delta t = 0.02$ and $h = 0.1$. The domain half-width is always set at $0.5 \times 7.5 \mu m \times 40 = 150 \mu m$. The total domain width is $300 \mu m = 0.3 mm$ which is comparable to the diameter of the tip of a ball point pen. The `maxNodeCount` used is 500 and the dislocation radius is $\delta = 0.01$. The summary statistics are taken at 600 sample times steps. Note that each run of the simulation takes around 7-10 minutes on an MSI laptop with a 12th Gen Intel(R) Core(TM) i7-12650H processor and a Nvidia GeForce RTX 3050 6 GB Laptop GPU. Using multi-threading via MATLAB's `parpool('threads')` and a `parfor` loop, 6 workers are employed for ensemble averages.

Throughout all runs, the values $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$ and $\lambda_4 = 0.5$ were kept fixed. The reason for these choices are summarised in table 4.1. That left λ_5 , λ_6 and λ_7 to be experimented with. In order to maintain a reasonable scope, only two numerical experiments were carried out in this work, however this is perhaps only scratching the surface in terms of interesting tests that could be done with `CellColonySimulator`. The two numerical studies carried out are detailed in sections 4.1 and 4.2.

Parameter choice	Reason
$\lambda_1 = 0.1$	Diffusion constant small to maintain numerical stability, but large enough to see appreciable spreading.
$\lambda_2 = 5.0$	Elasticity needed to be big enough to enforce the cell nominal length without being too stiff.
$\lambda_3 = 5.0$	Repulsivity made large and comparable to elasticity to enforce cell-cell exclusion as much as possible.
$\lambda_4 = 0.5$	Repulsion radius is set equal to cell nominal semi-major axis length.

Table 4.1: The rationale behind the selection of $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$

4.1 First numerical experiment: *Compactness verus cell aspect ratio*

In the first numerical experiment, we vary the cell aspect ratio, λ_7 , whilst maintaining $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, and $\lambda_6 = 1.0$. The goal is to explore how the cell aspect ratio, λ_7 , affects the overall colony compactness $C(t)$. The values of the mobility, λ_5 , and metabolic rate, λ_6 , are set to a standard value of 1.0, however, a comprehensive future study should investigate how these affect the relationship between compactness and cell aspect ratio.

In figure 4.6, the average compactness over time is plotted in green for cell aspect ratios of 0.4 (elongated), 0.5, 0.7 and 1.0 (circular). The yellow filled in area represents the 1.0σ confidence interval for the sample size of $N_{\text{ensemble}} = 6$. This means that, running the simulation at random, we would be approximately 68.2% sure that the compactness would lie in this range. Of course, this confidence interval could be tightened by taking a larger ensemble size, $N_{\text{ensemble}} \gg 6$ if supercomputing resources are available to the user. The value of $N_{\text{ensemble}} = 6$ is chosen here due to the MSI laptop used having 6 workers, but a larger value could be selected even on this hardware at the cost of multiplying the run time.

As expected, the compactness more or less increases over time for all simulations, however figure 4.7 shows that the ensemble average compactness, $\bar{C}(t)$, grows exponentially for higher aspect ratios (approaching circular), and for lower aspect ratios it seems to taper off and reach a local maximum in the case of $\lambda_7 = 0.4$. This suggests the counter-intuitive conclusion that, for the parameter selection given, more circular cells derive a more branched colony. This is evinced in figures 4.8 and 4.9, which shows the

colony itself (as insetted plots) as the compactness increases for $\lambda_7 = 1.0$ and $\lambda_7 = 0.5$, respectively.

In figure 4.8, at $t = 35$ hours, the compactness exceeds 20.0 and the colony takes on a branched shape. Conversely, in figure 4.9, the compactness never exceeds 10.0. In fact, a circular shaped colony with comparatively lower compactness appears to emerge from the overall dynamics which is a surprising finding. This appears to be supported by the ensemble averages shown in figure 4.7, in which colonies with lower aspect ratio ultimately reach a lower compactness perhaps due to emergent circularity.

This seems to go against the linear logic that a higher aspect ratio indicating circular cells should lead to a more circular colony. The paradox can be resolved when we interpret the model in terms of the nominal area of each cell. The area of a circle of radius $0.5 \times 7.5 \mu\text{m}$ is about $44.2 \mu\text{m}^2$, whereas the area of an ellipse is $\pi pq = \pi \times 3.75 \mu\text{m} \times 0.4 \times 3.75 \mu\text{m} = 17.7 \mu\text{m}^2$. The ratio of the areas is precisely λ_7 (0.4 in the most elongated case), which means that the circular cell is 2.5 times the area of the elongated cell. Note that we are talking about the fully grown size which I have previously characterised by the nominal cell length $L_0 = 7.5 \mu\text{m}$.

Interpreting this biologically, larger cell areas consume more nutrient leading to a nutrient deficit which induces a more branched growth regime having higher compactness. Conversely, lower cell areas consume less nutrient leading to increased crowding and the emergence of colony scale circularity.

Another observation which suggests the need for further experimentation is the qualitative difference between the ensemble average compactness of $\lambda_7 = 0.4, 0.5, 0.7$ and 1.0 demonstrated in figure 4.7. The growth curve for $\lambda_7 = 1.0$ appears to be roughly exponential, whereas, the growth curve for $\lambda_7 = 0.4$ is almost logarithmic. How we classify the curves qualitatively is arbitrary, but the comparison at the very least points to a qualitative difference between the two ends of the scale of λ_7 . A question that could now be posed to biologists is whether there is a critical value at which this transition occurs. Given supercomputing time, this model may indeed produce a hypothesis regarding a critical value of λ_7 , which could be tested experimentally. It may also be the case that over a large enough time scale we would see branching behaviour even for low values of λ_7 which would indicate two turning points: a local maximum followed by a local minimum after which the compactness increases at an increasing rate. This “double-ascent” if it were found could also be operationalisable.

4.2 Second numerical experiment: *Growth rate versus mobility*

In the second numerical experiment, the growth rate $\mu(t)$ was investigated with respect to its dependence on the mobility, λ_5 . In all runs, the growth rate was found to be decreasing during times when the number of cells was fixed. At times when new cells were added in, the growth rate would more or less instantaneously increase.

As evinced in figure 4.10, adding in new cells could result in what appears to be a resurgence in growth rate. Of course, further studies could focus on developing more natural ways to continuously add new cells into the simulation (see Chapter 5) which would allow for a smoother curve. The fact that cells are added in at discrete intervals is a limitation of the model.

Predictably, figures 4.12 and 4.13 demonstrate that the growth rate decreases more slowly for colonies with higher mobility, for instance $\lambda_5 = 2.5, 5.0$. Biologically, this is likely due to the fact that new daughter cells can move more quickly to positions of higher nutrient concentration due to a higher magnitude of chemotaxis. Conversely, colonies with lower mobility deplete nutrient at the same rate however they are unable to move away quickly enough to maintain a higher growth rate.

Once again, this study is somewhat limited by the hardware used. Given more compute power, a simulation across longer time scales and over larger ensembles is possible which could unveil more findings, in addition to making the current findings more conclusive.

4.3 Summary: *Uncovering different growth regimes*

Overall, the model has successfully demonstrated the ability to image different growth regimes of baker's yeast morphology. The primary strength of the model is its versatility: it can model yeast morphology in both a nutrient deficit (small λ_6) and a nutrient rich (large λ_6) case, wherein different qualitative behaviour is derived.

The hope is that this type of algebraic modelling of cell colonies can be generalised to cancer morphology. It is my prediction that the model, when outfitted to a 3D setting with access to nutrient sources such as blood vessels could produce fascinating findings which could motivate further *in vitro* or perhaps even *in vivo* experiments. One possible application is to tumour spheroids. In this instance, the model could

be promoted to a 3D model which is not a conceptual leap but may prove to be a formidable computational task.

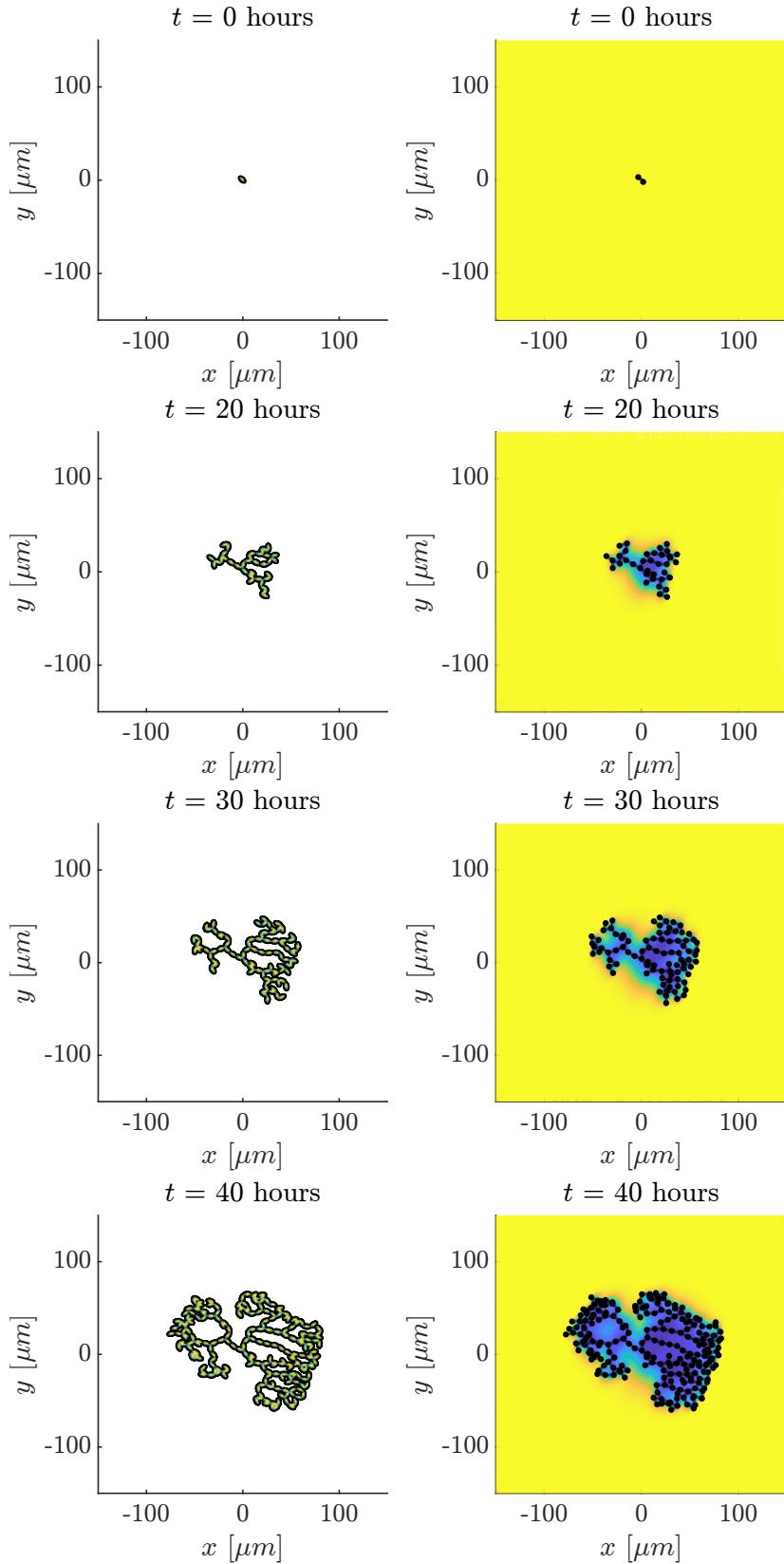


Figure 4.1: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 3.0$, $\lambda_7 = 0.5$. On the left we have the biomass field, the nutrient field is on the right.

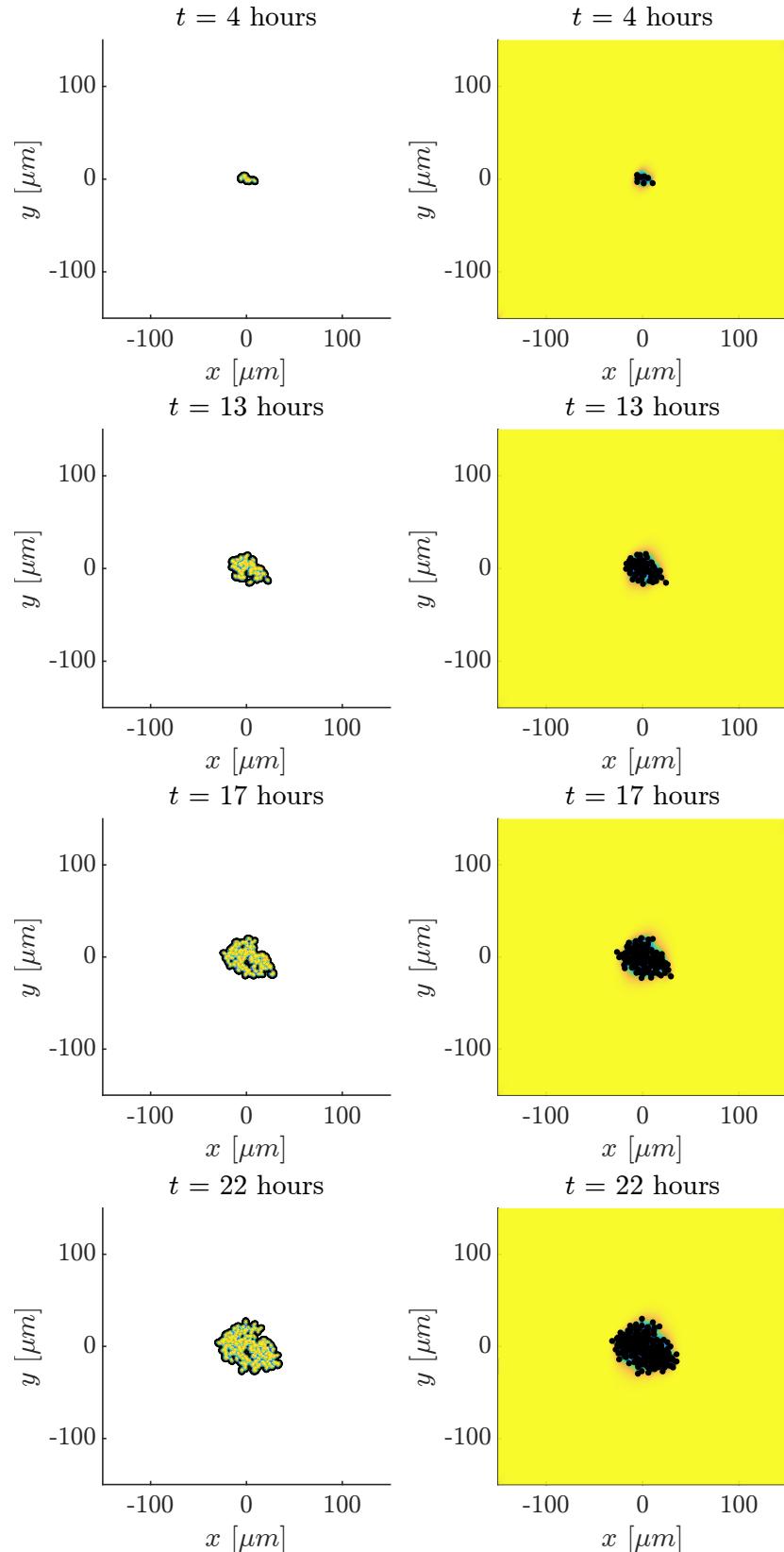


Figure 4.2: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 0.1$, $\lambda_6 = 0.7$, $\lambda_7 = 0.73$. On the left we have the biomass field, the nutrient field is on the right. The aspect ratio is $\lambda_7 = 5.5/7.5 = 0.73$.

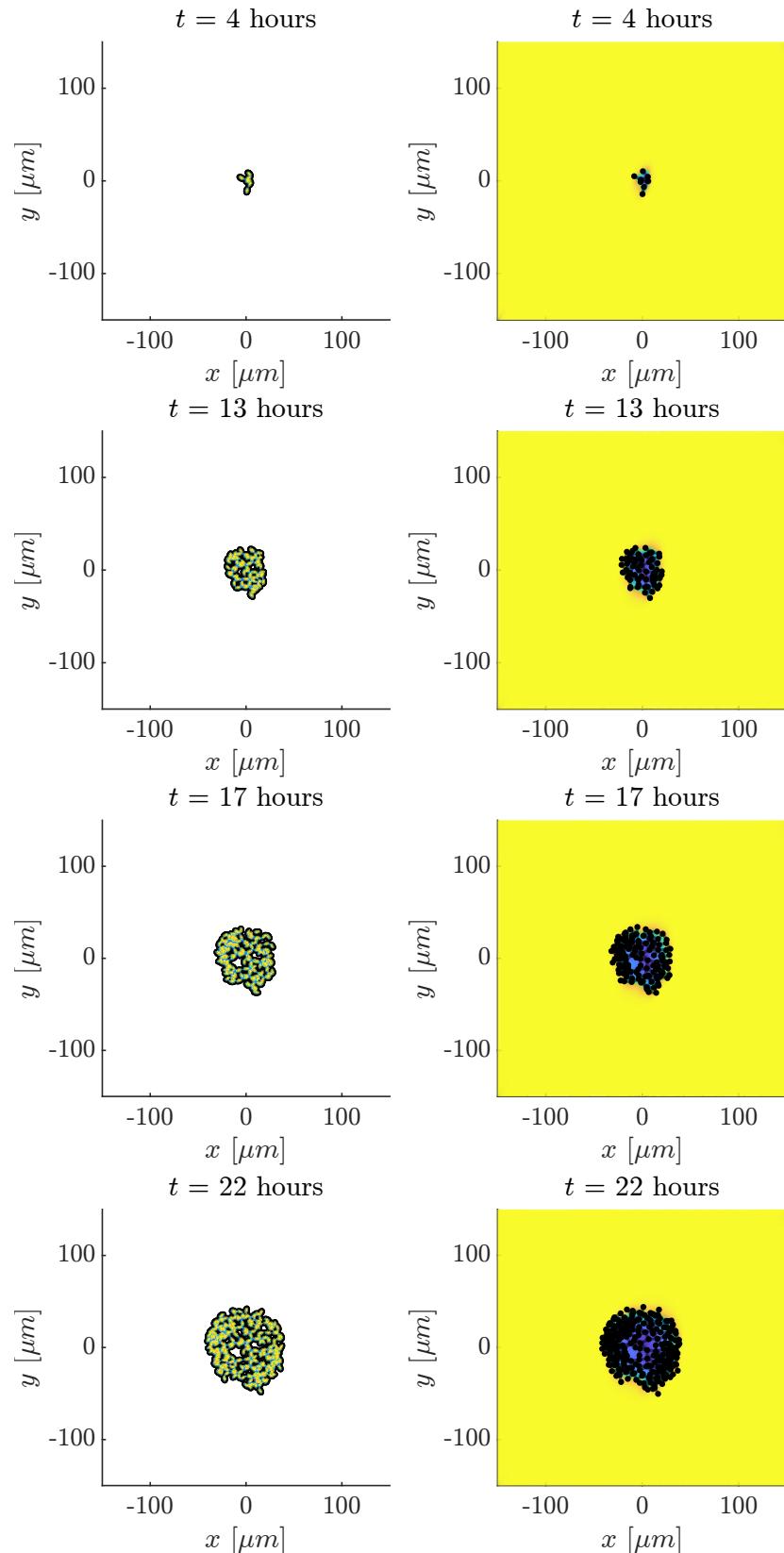


Figure 4.3: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.2$, $\lambda_6 = 0.7$, $\lambda_7 = 0.6$. Biomass on left and nutrient field on the right.

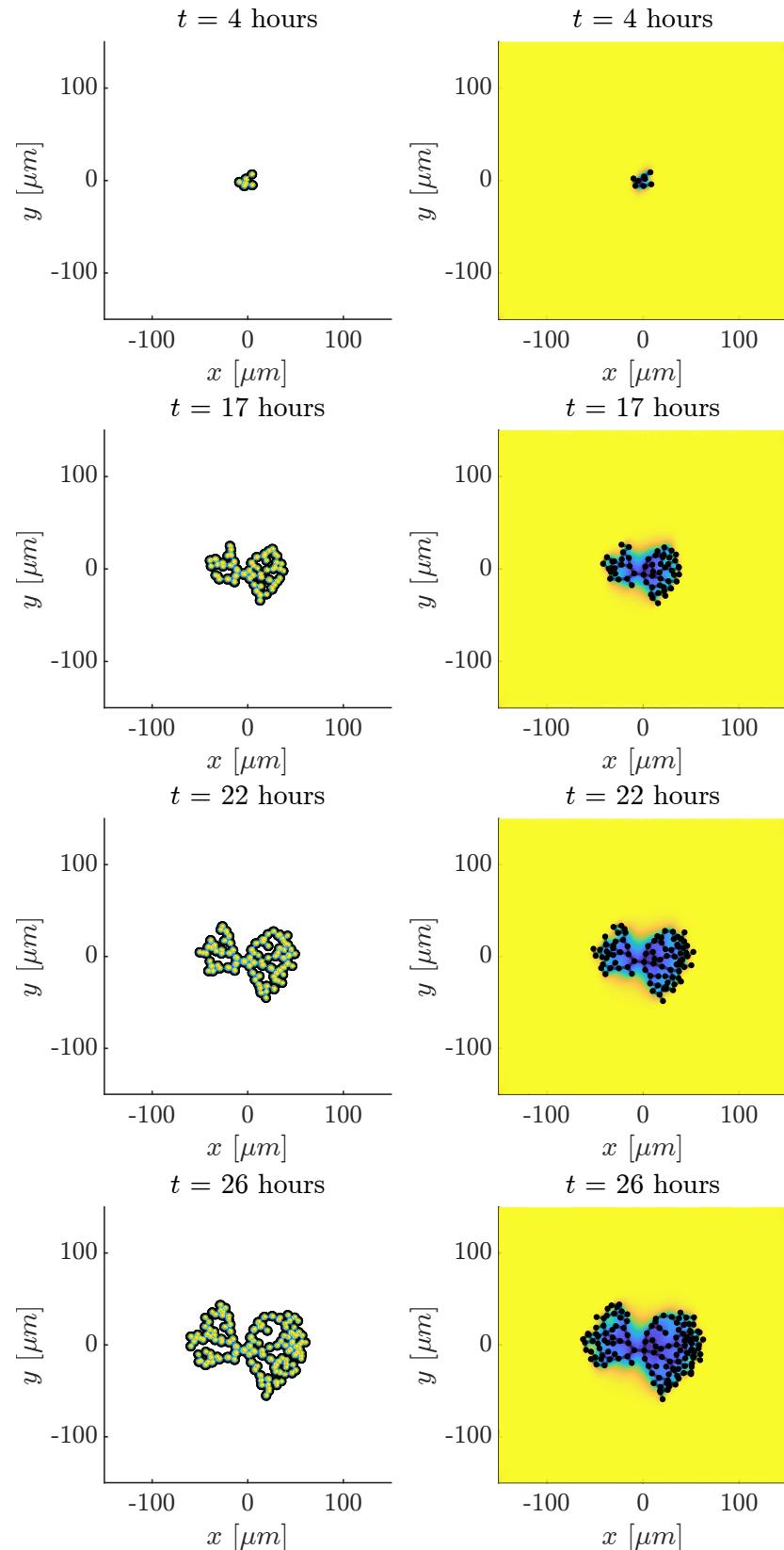


Figure 4.4: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.6$, $\lambda_6 = 0.9$, $\lambda_7 = 0.95$. Biomass on left and nutrient field on the right.

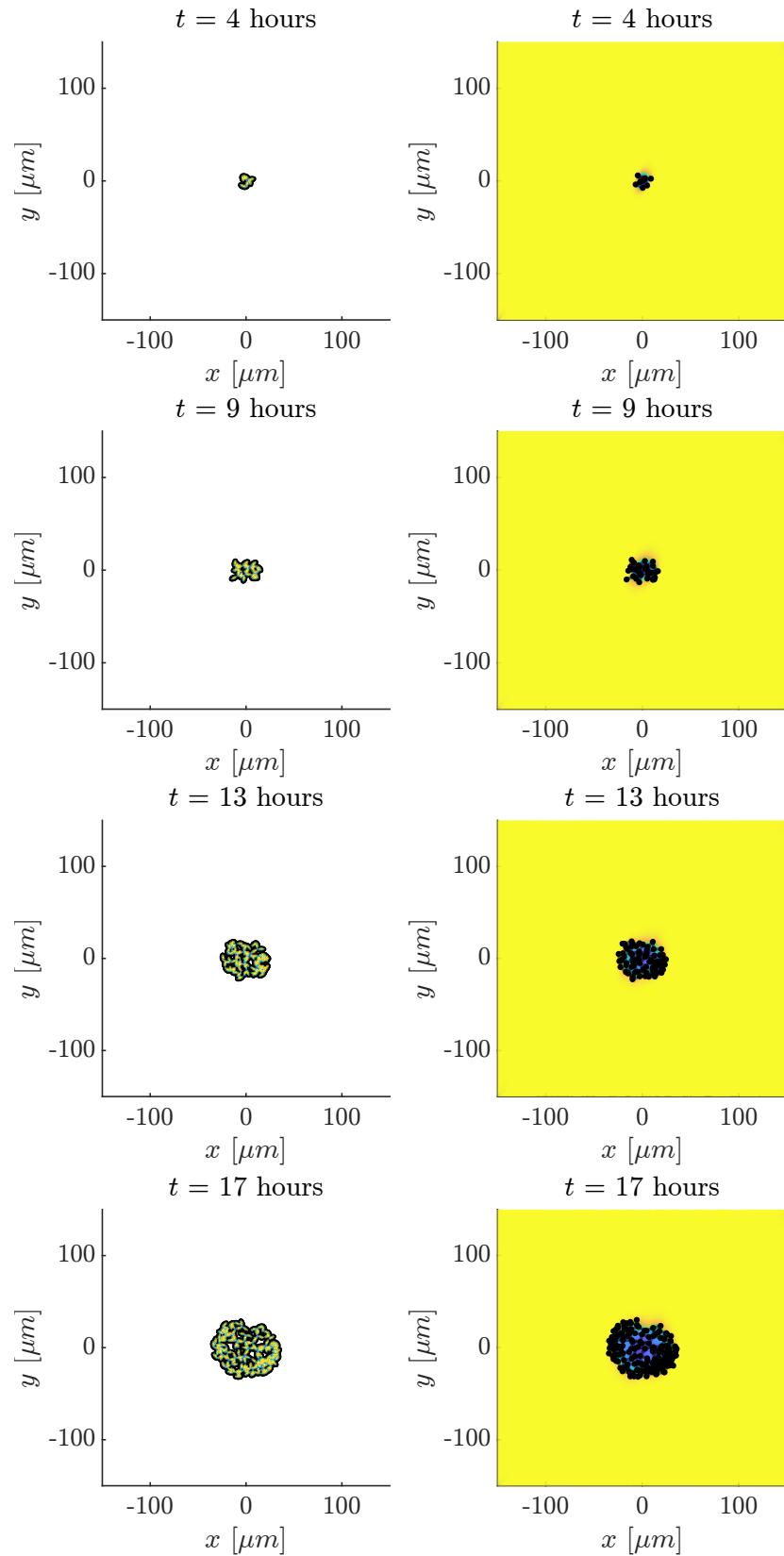


Figure 4.5: A cell colony with parameter values given by $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.6$, $\lambda_6 = 0.7$, $\lambda_7 = 0.5$. Biomass on left and nutrient field on the right.

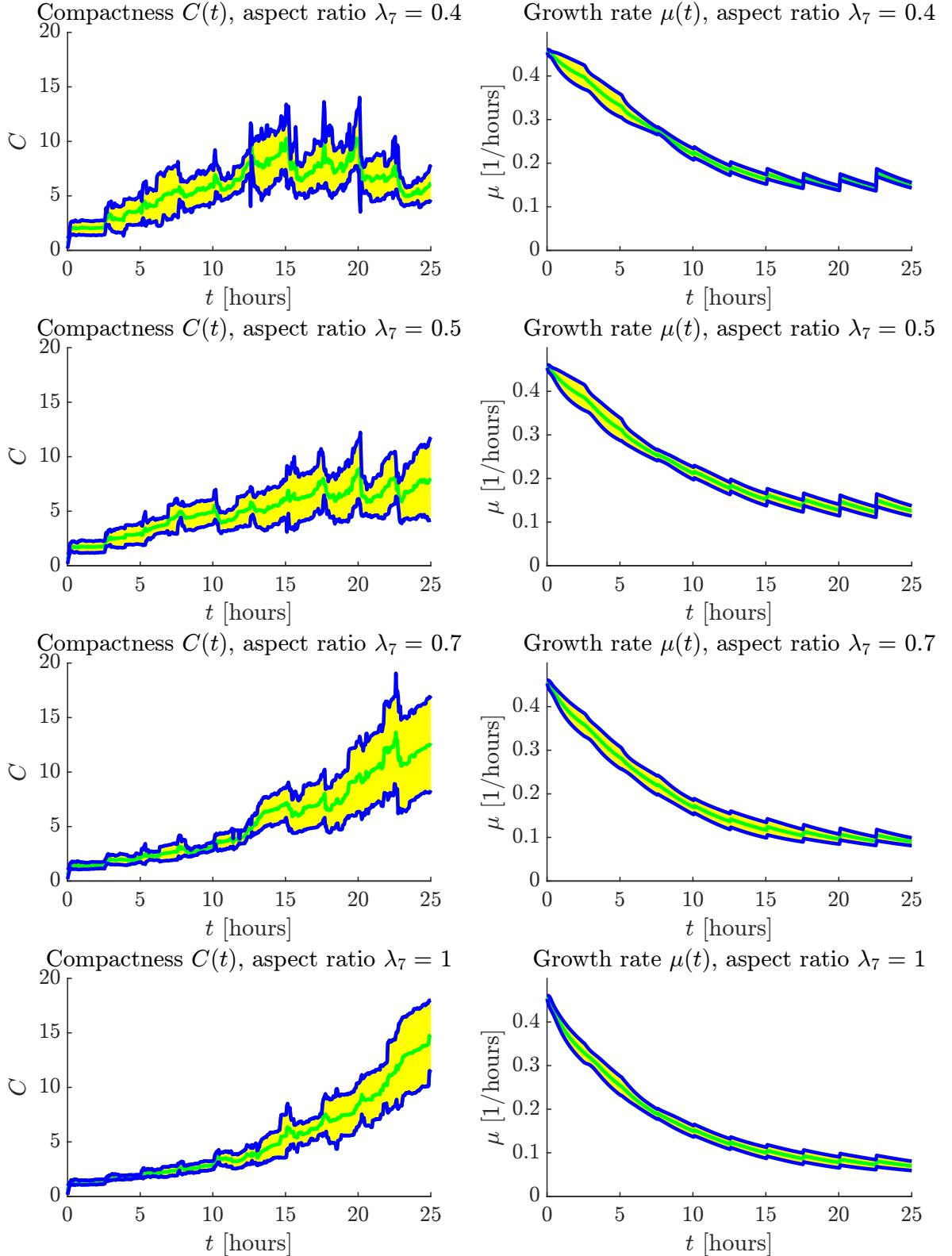


Figure 4.6: The colony compactness and growth rate for $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 1.0$, $\lambda_7 = 0.4, 0.5, 0.7, 1.0$ and an ensemble size of 6.

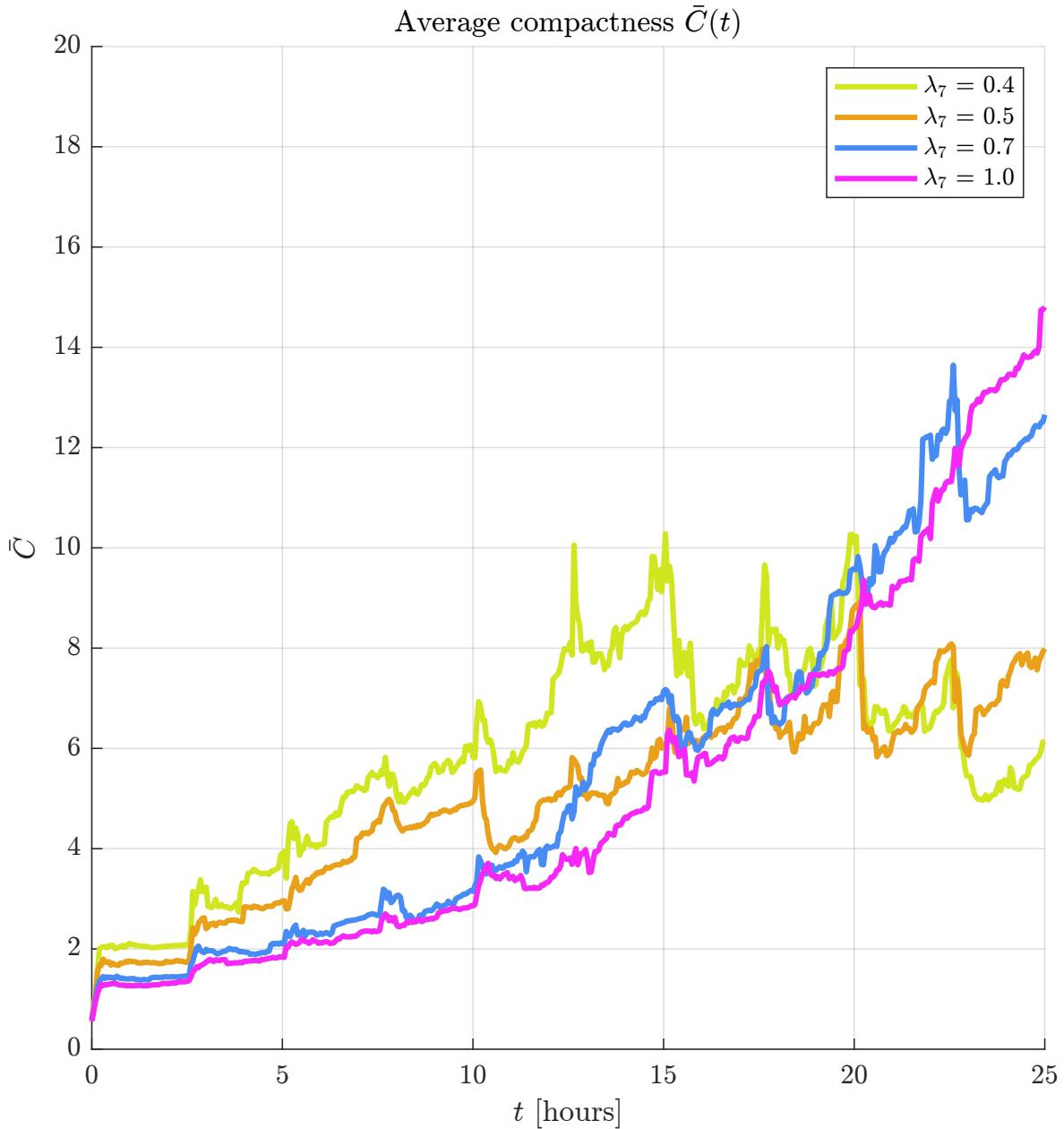


Figure 4.7: The average colony compactness for $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 1.0$, $\lambda_7 = 0.4, 0.5, 0.7, 1.0$ and an ensemble size of 6.

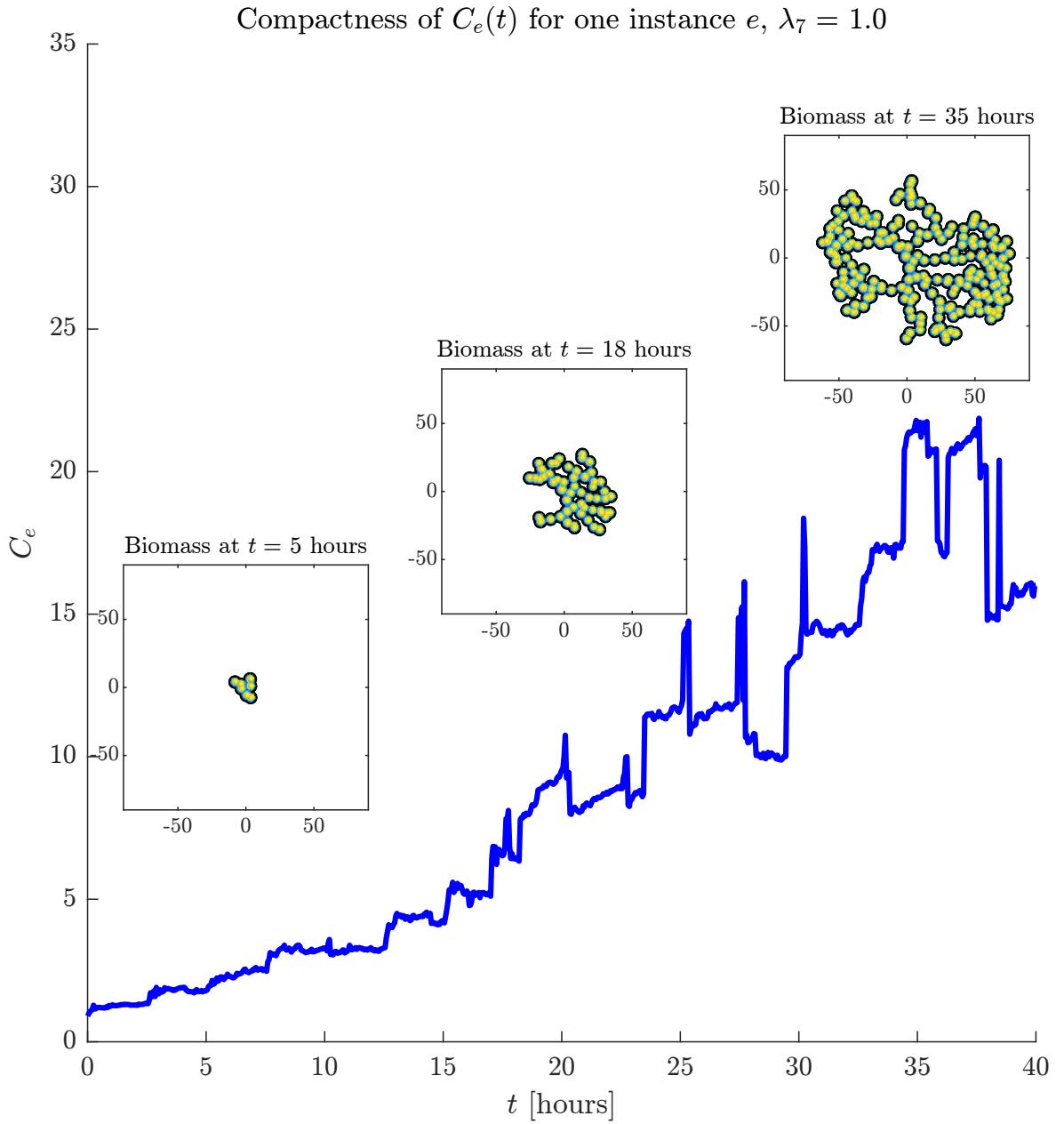


Figure 4.8: Compactness for ensemble instance $e = 1$ and $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 1.0$, $\lambda_7 = 1.0$. The inset plots show the biomass at three times for comparison with compactness.

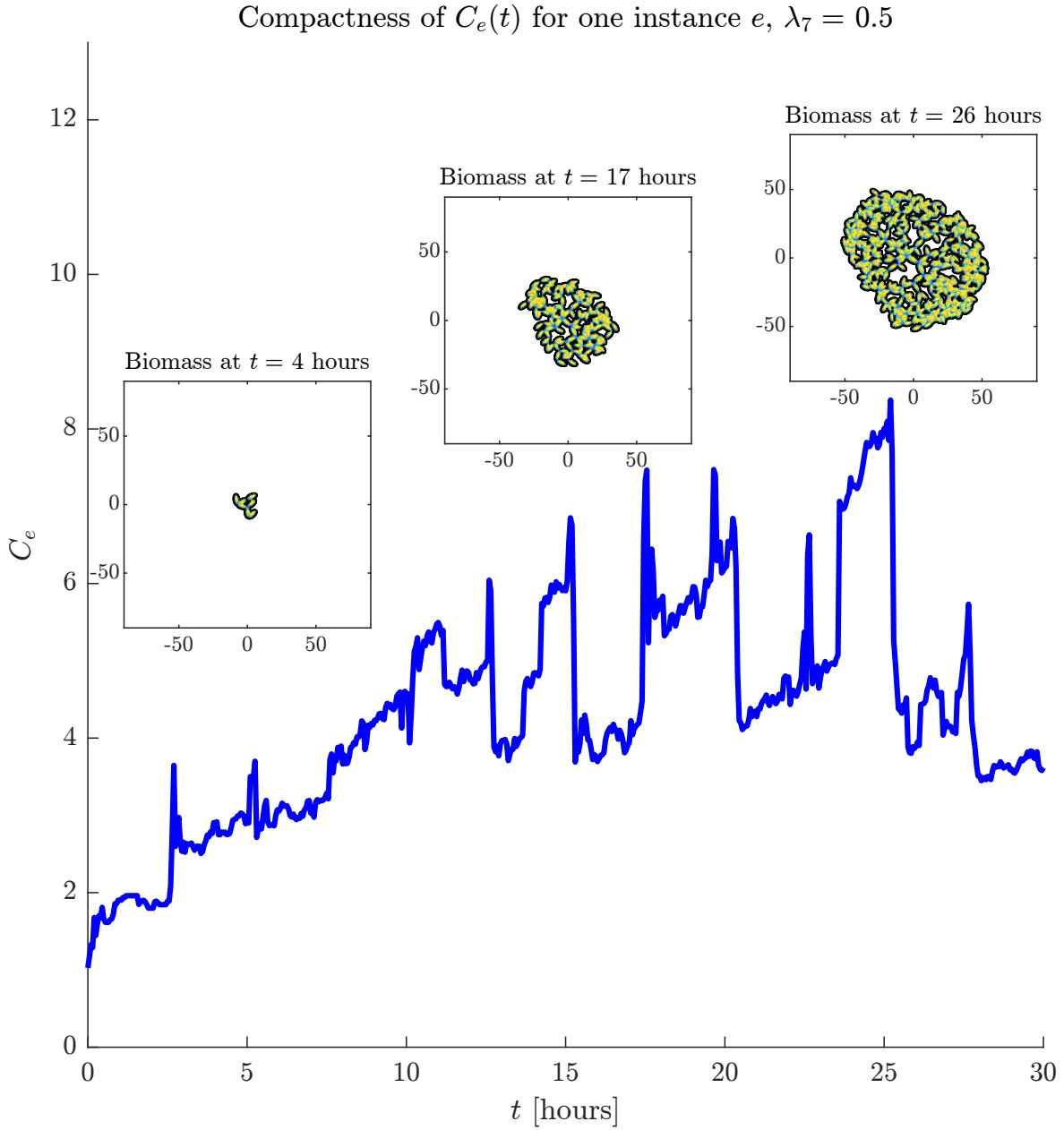


Figure 4.9: Compactness for ensemble instance $e = 1$ and $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 1.0$, $\lambda_6 = 1.0$, $\lambda_7 = 0.5$. The inset plots show the biomass at three times for comparison with compactness.

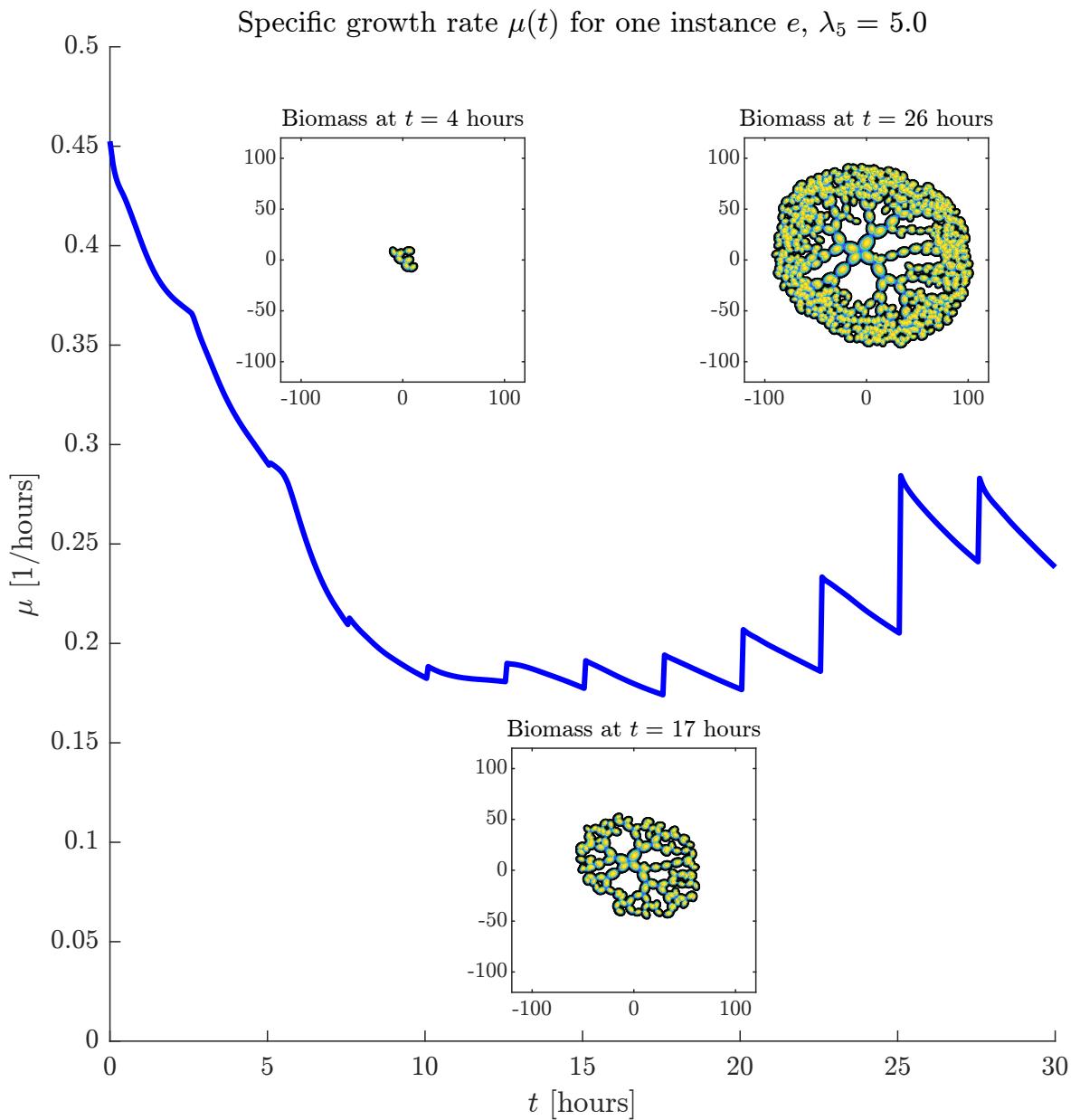


Figure 4.10: Specific growth rate $\mu(t)$ for ensemble instance $e = 1$ and $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 5.0$, $\lambda_6 = 1.0$, $\lambda_7 = 0.7$.

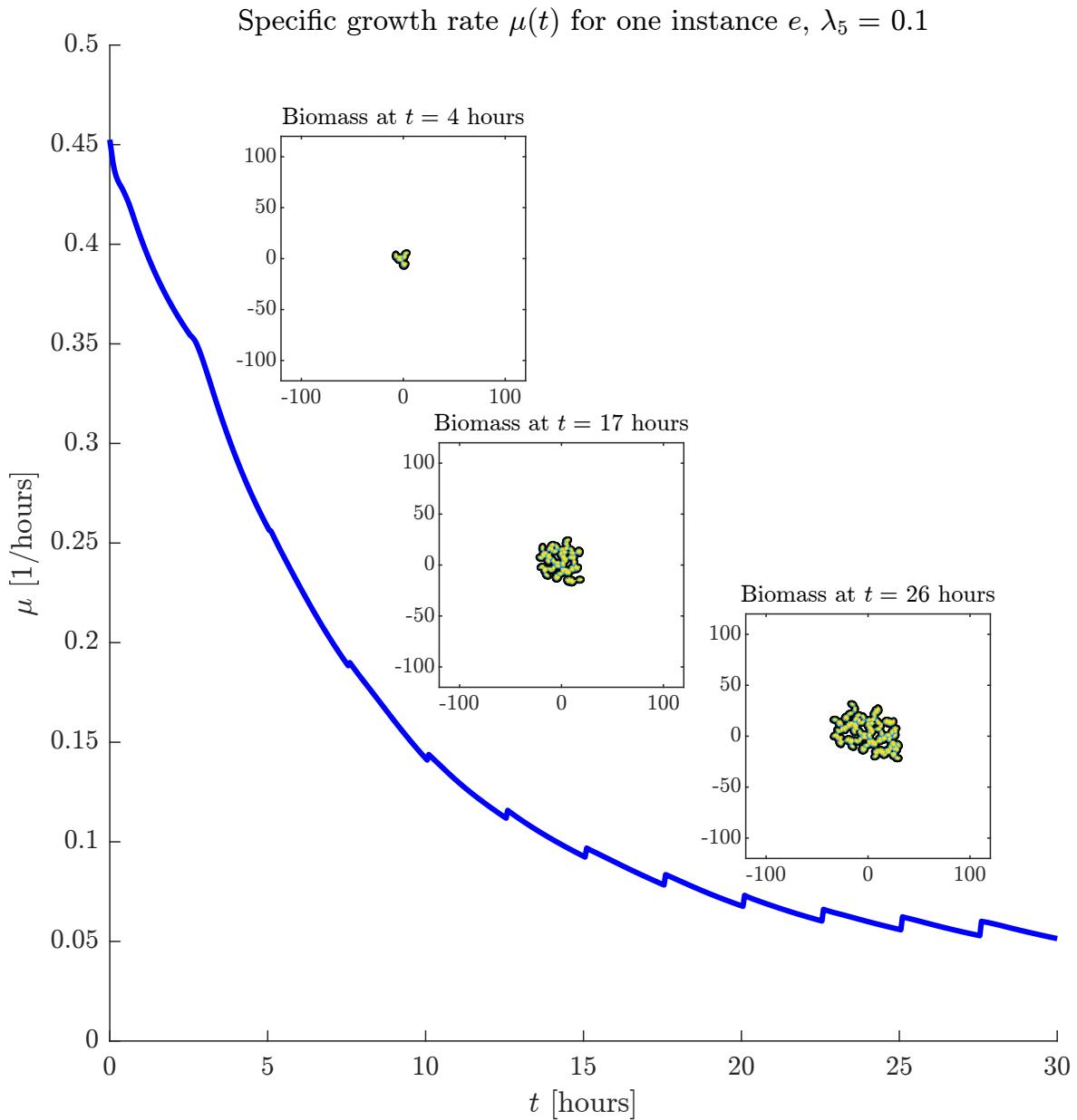


Figure 4.11: Specific growth rate $\mu(t)$ for ensemble instance $e = 1$ and $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 0.1$, $\lambda_6 = 1.0$, $\lambda_7 = 0.7$.

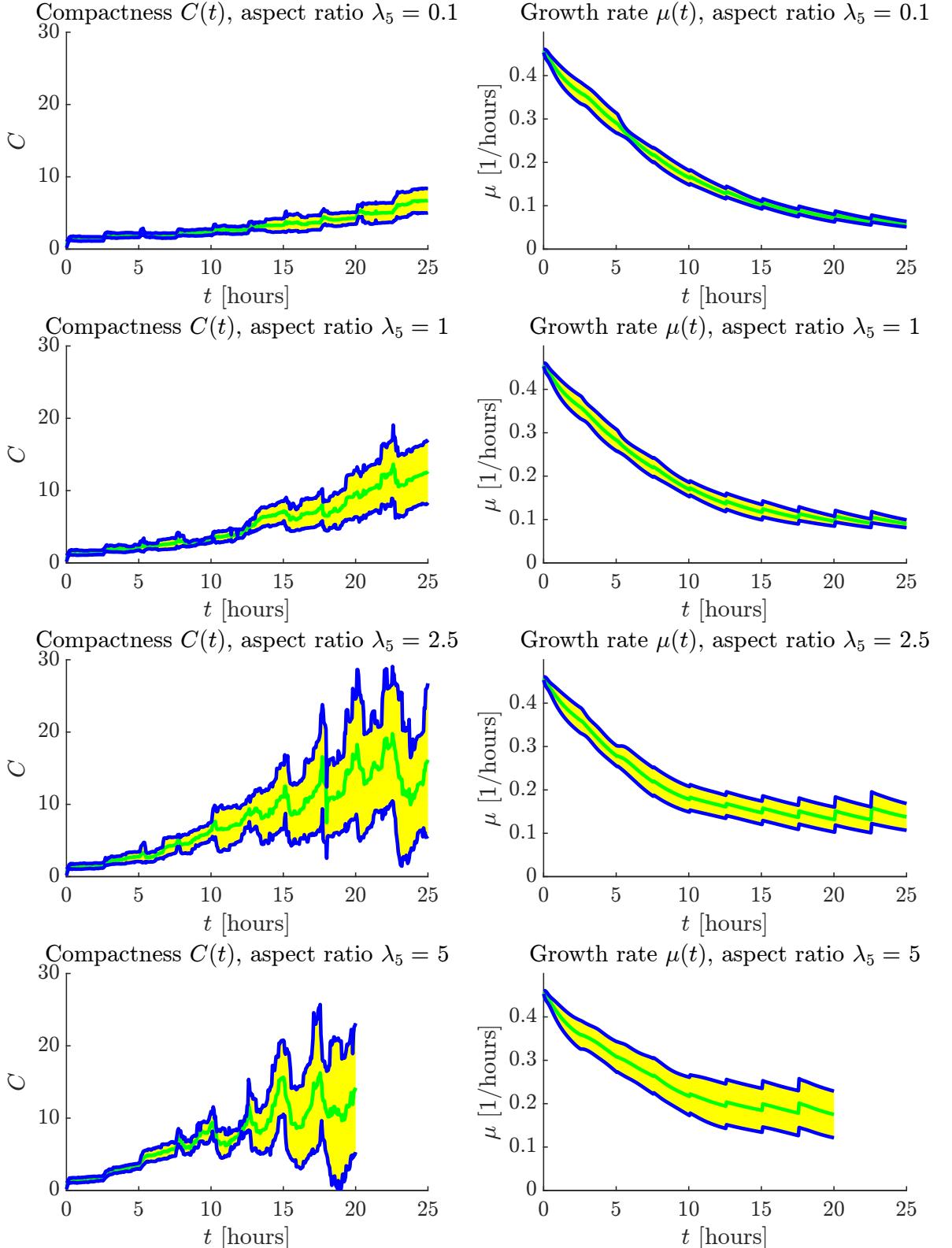


Figure 4.12: Compactness $C(t)$ and specific growth rate $\mu(t)$ for $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 0.1, 1.0, 2.5, 5.0$, $\lambda_6 = 1.0$, $\lambda_7 = 0.7$, and an ensemble of size 6. Note that the bottom panels could only be simulated to $t = 20$ hours due to GPU out-of-memory issues.

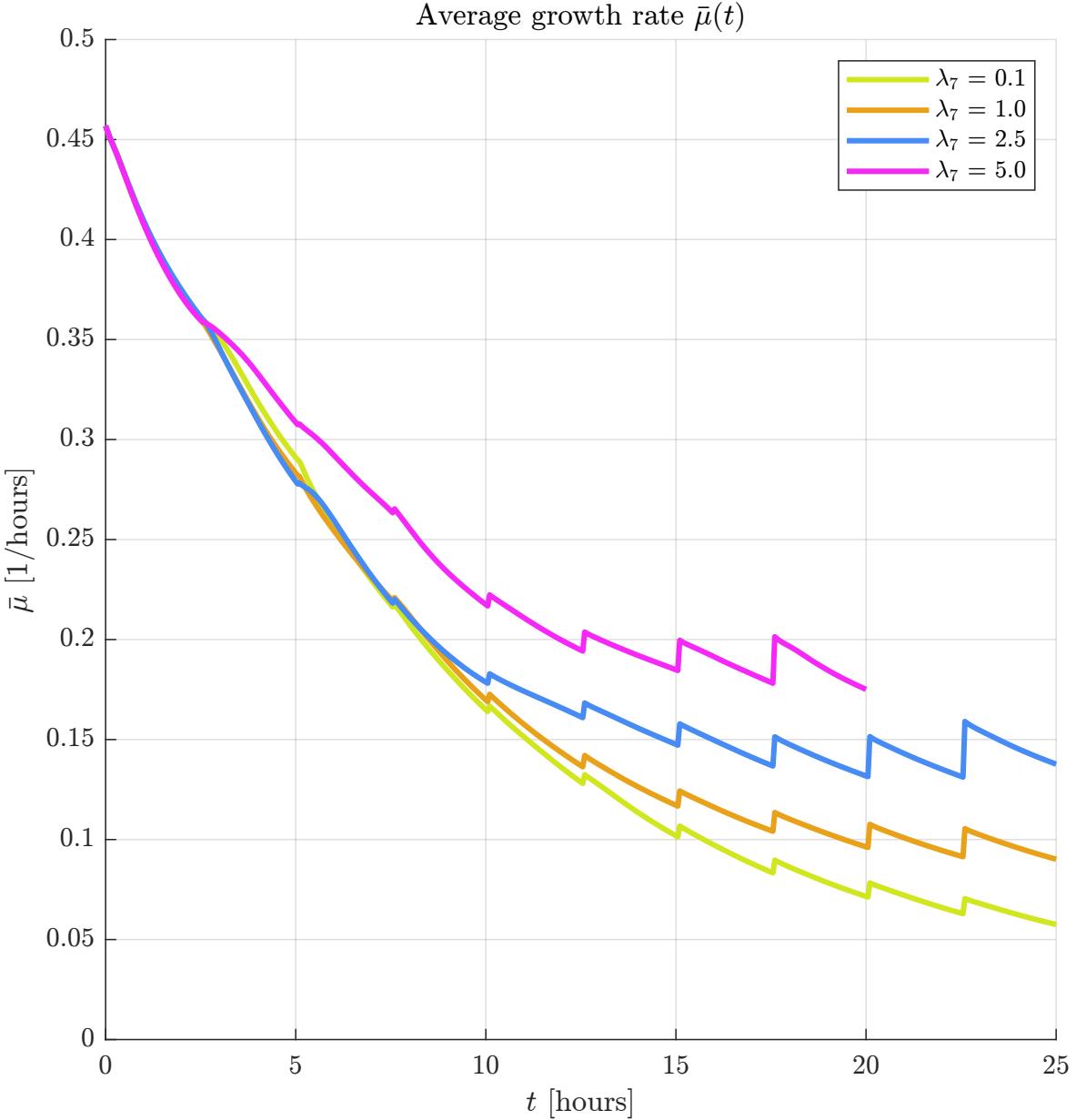


Figure 4.13: Average growth rate $\mu(t)$ for $\lambda_1 = 0.1$, $\lambda_2 = 5.0$, $\lambda_3 = 5.0$, $\lambda_4 = 0.5$, $\lambda_5 = 0.1, 1.0, 2.5, 5.0$, $\lambda_6 = 1.0$, $\lambda_7 = 0.7$, and an ensemble of size 6. Note the plot for $\lambda_5 = 5.0$ could not be completed on the available hardware due to GPU out-of-memory.

Chapter 5

Discussion and limitations

5.1 Evaluation of the model

The computational model **CellColonySimulator** incorporates design decisions based on a number of concerns. These include

- The requirement for manageable run times on a gaming laptop.
- Modelling both colony scale and cell scale morphology.
- Capturing the process of mitosis in a natural but simple way.
- Ensuring the model is amenable to statistical analysis.
- Providing a model ready for fitting to experimental micrographs of cell colonies.
- Crystallising the model into a form that could be analysed in a conceptually rigorous setting, say within the context of functional analysis, bifurcation theory or algebraic topology.

Whilst the computational model and conceptual apparatus developed here take on these concerns, this section will explore to what extent these concerns were *not* met. In particular, I comment on the fundamental limitations of the model. In subsection 5.1.1, the commentary begins with an evaluation of the model regarding the primary conceptual goals.

5.1.1 Conceptual limitations

The model was able to deploy the concept of algebraic curves with time dependent coefficients to represent the cells. Indeed the implicit equation for a translated, rotated, and scaled ellipse is a member of the ring of polynomials in two variables $\mathbb{R}[x, y]$ where

coefficients are dependent on the underlying discrete network of nodes. In a previous version of the model, the underlying network of nodes were initially collocated and via the elimination of constraints the network was able to unfold over time. This was implemented with logical “mask” matrices in MATLAB, but brought practical problems. One such problem was that branches could easily deplete their store of nodes, and indeed the number of nodes chosen at the beginning of the simulation had to be arbitrary. This meant that the previous model had prohibitive limits on the number of branches and the size of the branches. Another limitation, was that the ODE solver was slowed down by the need to simulate the evolution of collocated vertices.

CellColonySimulator overcomes this issue by simply adding new nodes to the network, just very close by ($\delta = 0.01L_0$). As it turned out, this was not a consequential fault, since the distance δ can be made so small that the daughter cell appears to bud from the parent cell, as would be observed in the case of pseudohyphal growth of baker’s yeast. What I argue is that this is in fact a conceptual fault present in the model because it represents an arbitrary addition of a new variable, the budding angle θ_b not to mention the constant (but small) budding radius δ . What I intended was a “derivation” of cell division within a broader model, as opposed to an imposition of more assumptions and parameters over time to achieve the mitosis.

Unsatisfyingly, the budding angle is chosen to be uniformly distributed on $[0, 2\pi]$ and is the only source of randomness in the model. Of course the budding angle could be chosen to be in line with the nutrient gradient vector and this would eliminate randomness from the model. Future work could seek to reintroduce randomness in the form of a stochastic process, such as a Langevin term in the ODEs for the node positions (and even in the nutrient PDE).

A model which I have prototyped, is one wherein the nutrient field is effectively “converted into nodes”. That is, the nutrient field is not simply a scalar field of “stuff”, but actually becomes a field of structure. The simplest version of this is to set the nutrient field as a collection of Brownian walkers that can be taken up by the colony and added to the colony’s network. In some sense this would solve the problem, since no new nodes are ever added into the simulation, it is just that the initially unstructured nodes take on a new behaviour within the cell network.

Another conceptual direction is that of taking seriously the notion of “cell as structured randomness”. Biochemistry tells us that cells are reducible to chemical reaction networks (CRNs), but of course to combine CRNs into the workings of a single cell is not trivial. This means effective models are required, which leave out most of the labyrinthine details of the cell’s internal workings (such as proteins, DNA, RNA, mitotic

spindles, golgi apparatus) but allow for efficient computation. Even CRNs are limited by the fact that they usually do not consider the shape of a protein which plays a key role in its functionality. The model presented here brushes over all of these details in favour of an effective mathematical approach, but future work could focus linking nanoscopic molecular scale and microscopic cellular scale.

Regarding the shape of individual cells, the ellipse with variable aspect ratio is chosen to represent eukaryotic fungal cells such as *S. cerevisiae*. Beyond changing the cell aspect ratio, there is no mechanism in **CellColonySimulator** to opt for plant like cell shapes (for example) which are more blocky. Algebraic modelling of the cell membrane (in the case of plant cells) or modelling of other cell wall shapes (fungal or even mammal cells) is implied here, however. For instance, by choosing more elaborate polynomials, one can derive a taxonomy of cell shapes at the cost of implementation simplicity and perhaps computational performance. An avenue to explore is seeing whether bifurcations can be achieved in a different ring of continuous functions of x and y . Indeed, can the whole colony be represented by (the level section of) one function at each time, such that the growth of the colony can be given by a homotopy between two functions?

Philosophically, 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? 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 and a nutrient field. Indeed, their coupling determines the dynamics.

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 of *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 development are somehow contained within the structure (for instance, DNA). It is 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.

Physically, there are again more directions to follow up. Consider a minimalistic example of modelling the motion of a spherical ball rolling around on a table. Two spatial parameters x and y , together as a tuple are enough to say where it is. However, if the ball unexpectedly 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.

Regarding the possibilities of geometric modelling, 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 systems in which a single cell can have nontrivial topologies, see dictyostelid cellular slime molds, [21]. 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 biological morphology viewed in the framework of “growing geometry”.

5.1.2 Dynamical limitations

A (fixed node count) cell colony undergoes the dynamics of an overdamped spring network capable of internal collisions driven by an external nutrient field. The key concern is whether this dynamical structure is grounded in biological evidence, and whether the model is just metaphorical. Firstly, it is worth noting that because we are in the overdamped regime, there are no intrinsic oscillations between connected nodes. This is because the node velocity is proportional to the force as opposed to the case of an underdamped harmonic oscillator, where the acceleration is proportional to force. In answer to the concern posed, the current model is an effective one however it is feasible that the model can be fitted to experimental micrograph data, which is commented on in section 5.2, and thus the seven input parameters could be found from real data.

The springs have a neutral position which is a cell length distant from the parent node. In other words, the spring force encodes a passage between equilibrium positions of the colony. The presence of an (external) chemotactic force coupled into the dynamics complicates this by gradually moving the equilibrium position for a fixed node count colony outwards. One observation of the simulation results, was that internal cells grew beyond the nominal cell length due to the additional force of chemotaxis. In practice, this issue is alleviated by increasing the cell elasticity, λ_2 , but this is nonetheless a limitation.

New nodes are added to the network nearby old nodes as stated in Chapter 2. The timing of the addition of new nodes is somewhat adhoc and provisional, but could in theory be elevated to something more sophisticated. Starting with the colony growth rate $\mu(t)$ which is taken to be a fraction of the ideal growth rate $\mu^* = 0.4 \text{ hour}^{-1}$, cells are added in at discrete intervals of time, given by

$$\Delta n = \left\lceil \frac{1}{\Delta t} \right\rceil.$$

The value Δn is the number of time steps between two (global) mitosis events. For a time step of $\Delta t = 0.02$, this amounts to $\Delta n = 50$ time steps. The amount of nodes added at this time step, denoted n^* (a multiple of Δn), is given by $(e^{\mu(t_{n^*})} - 1)N_{\text{nodes}}(t_{n^*})$ where $\mu(t_{n^*})$ is the average value of the nutrient field (recall that the nutrient field is assumed to be 1.0 at the simulation outset). The assumption that growth rate is defined in terms of the nutrient field makes sense, however the assumption of it being *globally* defined thus is somewhat dubious. A possible improvement is to draw up a coarse grid in which each square can fit a dozen or more cells, and say that the growth rate $\mu = \mu(x, y, t)$ is locally defined by averaging nutrient per grid square. Of course, the reason that a spatially independent growth rate, $\mu = \mu(t)$, was chosen was for implementation simplicity.

The periodic structure of adding cells in at simultaneous snapshots of time, is an artificial choice based on implementation simplicity alone. A higher fidelity model could be constructed in which cells bud off at any time. One way to do this is to calibrate mitosis times per cell based on cell cycle data and allow a cell to undergo mitosis if a certain threshold of nutrient is locally available. This would not be difficult to implement, but comes with a few conceptual caveats. One issue is that a (discrete) logic would need to be introduced per cell, which opens up a fascinating area of study (“cell as computer program”), however this is outside the scope of this work.

Another limitation of the dynamical model is overly simplistic collisions. The collision mechanism is node based, as opposed to edge based or even SDF based. An attempt to improve upon the expedient hard ball collision scheme, is developed in Appendix A.2. Therein, constrained dynamics techniques are employed to ensure no pairwise overlap of the biomass level sections. It is clear that constrained dynamics is the way to go, but my unique implementation is not computationally efficient and therefore was not used in **CellColonySimulator**.

Finally, the use of a reaction-diffusion PDE is traditional in mathematical biology, however in light of the insights developed here, future work could focus on modelling

random fluctuations in nutrient concentration, or even introducing multiple interacting morphogens such as in Turing's morphogenesis paper [44].

5.1.3 Computational consolations and cell death

"Now it is autumn and the falling fruit and the long journey towards oblivion.... Have you built your ship of death, O have you? O build your ship of death, for you will need it."

An excerpt from *The Ship of Death*, by D.H.Lawrence.

I had originally written a MATLAB script called *removeCell.m* but I decided not to include this in **CellColonySimulator**. The reasons for apoptosis (the triggering of a cell death event) are probably very complex with respect to the whole organism. That being said it should be possible to analyse using statistical methods whether the cause is within the cell or within the whole colony, or both.

As far as computational efficiency and the choice of programming language is concerned, it was my initial intent to code **CellColonySimulator** in the C programming language using the structs of arrays (SOAs) data orientated approach.

In this direction, significant speedups can be made if skilled use of parallel threading and GPU compute (through CUDA) are employed. Visualisation of the colony could be achieved by converting the colony's implicit representation into a surface mesh and rendered on the GPU via a vertex shader program in OpenGL. Another alternative is to use ray marching where the object to be rendered is projected onto a quadrilateral arranged in the viewport. Rays projected forward from this quadrilateral locate the object via the object's SDF and lighting effects can easily be integrated. Because this can be done in a fragment shader in OpenGL it would provide a quick visualisation without much coding involved. The meshed solution, mentioned first, despite being more chunky, seems more flexible, because it can be integrated easily with other meshed-based scene elements, such as plot axes and labels.

Another approach, is to simulate the colony dynamics in a lower level language like C or Fortran, saving the node positions and nutrient field at each moment. The biomass can then be reconstructed from the node positions in a language like MATLAB or Python, where post-processing such as plotting and statistics can be done. The best of both words would be to toggle on or off OpenGL visualisation in C, to check a preliminary visualisation for any obvious issues, whilst saving the run data for post-processing in another language.

5.2 Readiness for experimental validation and fitting

Observing real yeast colonies, we note that multiple separate budding networks form. That is to say, the present model would need to be further modified to include detachment events where new cells form a separate disconnected colony.

The addition of robust cell death and cell detachment events would prime the model to be experimentally validated. This would likely introduce new parameters λ and more overall complexity, so an in-depth, flexible and robust modelling platform in C (as mentioned earlier) integrated with a good data analytics package would be ideal to carry out comparisons with experimental yeast micrographs at a manageable run time. The theory is that the highly optimised run script can be called upon from MATLAB or Python when required.

Statistical best fits to experimental morphological data could be derived by using metrics like compactness and average radius. The question becomes: *for what model parameter values can one minimise the difference squared between the simulated (ensemble average) and experimental images?* One could use a feed forward neural network (NN) in order to learn the model itself, modelling the output parameters, compactness and others, as a function of the input variables, λ .

Conclusions

5.3 Summary remarks

The components of this thesis when considered individually are adapted from previous mathematical modelling, sometimes in biology, sometimes not. The value of what has been done emerges from the interplay of the elements

1. A PDE nutrient field,
2. An overdamped and growing spring network,
3. A cell colony given as a level-section of a polynomial,

all of which I feel were necessary for the model's overall synthesis to occur. I by no means believe that even a small proportion of the total causes and effects have been accounted for in regard to baker's yeast proliferation in nutrient-poor conditions. But it is my hope that new model *triple-points* or even *multi-stable symbolic states* can be sought after in future studies.

Qualitatively different types of models, when coupled in this attentive way can give rise to new dynamics not found in any singular model. That being said, the process of learning and tending to more models than one can be laborious, if not grounded in some sort of overarching framework, even if that framework is to disappear when the total project emerges.

5.4 A word on my scientific process

The openness to testing different models without the promise of intellectual refuge in one prior form, came from my conviction that *life itself* is irreducible and heterogeneous. Our mathematical models sometimes reflect this truth, and sometimes they do not. In that sense, this thesis is a *personal reckoning* with the very structuring force

of our own assumptions. But there are other forces at play aswell.

The pressure of scientific tradition was a necessary influence on my project, and so was my own opposing reaction to its reductionist tendencies. Without these forces this project would not have branched out into what it became. Since I too am a living being whose cells are in constant interaction with the environment, it was necessary to bring my own subjectivity into this tango of symbols.

5.5 Philosophical remarks

The reader who is not philosophically inclined may ignore this.

O let us talk of quiet that we know, that we can know, the deep and lovely quiet of a strong heart at peace!

How can we this, our own quietus, make?

An excerpt from *The Ship of Death*, by D.H.Lawrence.

In classical mechanics, forces are vectors that are summed to give the net force. What we have come to forget is that individual forces themselves could have different mathematical structures associated to them. There may be more natural or ergonomic formalisms associated to each force. The direct approach then, is to consider “force formalisms” that allow for designed trajectories.

It seems, “change how you look at it, and you will see something different,” is a dismissal of form. That is not the case. There is a rich history of mathematics that studies subjectivity directly, but you can only see that if you sit with it for long enough. Mathematics is wedded to form and form is also about perception.

Like Lindenmayer systems, *symbolic ideas* that undergo mutations and dilutions often become what they are not. Is this a symptom of the nature of nature? I do not comment.

The next stage of my mathematical career awaits. Inspired not by a megalomaniacal control of *subjectivity*, but by a wish to cherish subjectivity’s mysterious forms by building mathematical ornaments for it. As far as attention is misused for prediction at the cost of its own attentive form, I cannot offer a conclusive remark.

Computation as a form of reified failure to predict in full the face of the *real* is our great human inebriation. It is a dream never fulfilled to reinvent the human being

as a machine, because the machine is co-emergent with human nature. The *dream*, however, is a wonderfully productive one.

Ironically then, for an applied mathematics thesis grounded in the observation of the organism *S. cerevisiae*, this work is indebted to what I imagine the subjects *topology*, *geometry*, and *algebra* to be like. I have studied these topics haphazardly and hope to refine my understanding over the next few years.

It appears two boats have parted at a *Y*-junction in a river. Perhaps one day they will meet again. Perhaps not.

Appendix

A.1 Collocation algorithm with mask matrices

Each cell indexed $j \in \{1, \dots, N\}$ 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 intercellular 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 $j \in \{1, \dots, N\}$ 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

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

where (x_n, y_n) is the coordinate of the n -th vertex for $n \in \{1, \dots, V\}$. Note that the change in the number of cells is simulated by removing constraints between the vertices.

At the beginning of the simulation $x_1 = \dots = x_V$ and $y_1 = \dots = y_V$. 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 $C(\alpha_1, \dots, \alpha_M) = \text{diag}(\mathbf{1}_{V/2^{\alpha_1}}, \dots, \mathbf{1}_{V/2^{\alpha_M}})$ 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, $C(\alpha_1, \alpha_2, \dots, \alpha_M) \rightarrow C(\alpha_1, \dots, \alpha_q + 1, \alpha_q + 1, \dots, \alpha_M)$. 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

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

A.2 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 [50] based on constrained dynamics. The constraint in this case is that the area C remains 0 for all times,

$$C(\mathbf{q}) = 0, \quad (1)$$

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 constituent 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

$$\mathbf{q}_j = \begin{bmatrix} x_j \\ y_j \\ \theta_j \end{bmatrix}. \quad (2)$$

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,

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

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

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

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:

$$\mathbf{q}_j(t) = \begin{bmatrix} x_j(t) \\ y_j(t) \\ \theta_j(t) \\ b_j(t) \\ R_j(t) \end{bmatrix}, \quad (3)$$

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

$$\Omega(\mathbf{q}) = \{(x, y) \in \mathbb{R}^2 \mid f_{\text{intersect}}(x, y, \mathbf{q}) \leq 0\}, \quad (4)$$

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

$$C(\mathbf{q}) = \int \int_{\Omega(\mathbf{q})} dx dy. \quad (5)$$

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

$$C(\mathbf{q}) = \sum_{i \in SCC(\mathbf{q})} \int \int_{\Omega_i(\mathbf{q})} dx dy. \quad (6)$$

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

$$\frac{d}{dt} \int \int_{\Omega_i(\mathbf{q})} dx dy = \int \int_{\Omega_i(\mathbf{q})} \frac{\partial(1)}{\partial t} dx dy + \int_{\partial\Omega_i(\mathbf{q})} (1) \mathbf{v}_{\partial\Omega_i(\mathbf{q})} \cdot \hat{\mathbf{n}} dl, \quad (7)$$

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.

$$C(\mathbf{q}) = \int \int_D \left[\frac{1}{2} + \frac{1}{2} \tanh \left(-\frac{1}{K} f_{\text{intersect}}(x, y, \mathbf{q}) \right) \right] dx dy, \quad (8)$$

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

$$\frac{\partial C(\mathbf{q})}{\partial \mathbf{q}} = -\frac{1}{2K} \int \int_D \left[1 - \tanh^2 \left(-\frac{f_{\text{intersect}}(x, y, \mathbf{q})}{K} \right) \right] \frac{\partial f_{\text{intersect}}(x, y, \mathbf{q})}{\partial \mathbf{q}} dx dy. \quad (9)$$

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

$$J_\beta = -\frac{1}{2K} \int \int_D \left[1 - \tanh^2 \left(-\frac{f(x, y, \mathbf{q})}{K} \right) \right] \frac{\partial f(x, y, \mathbf{q})}{\partial q_\beta} dx dy. \quad (10)$$

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 .

$$\frac{\partial^2 C}{\partial q_\alpha \partial q_\beta} = \frac{\partial}{\partial q_\alpha} J_\beta \quad (11)$$

Crunching the calculations we get

$$\frac{\partial}{\partial q_\alpha} J_\beta = -\frac{1}{4K^2} \int \int_D \left[1 - \tanh^2 \left(-\frac{f}{K} \right) \right] \left[\tanh \left(-\frac{f}{K} \right) \frac{\partial f}{\partial q_\alpha} \frac{\partial f}{\partial q_\beta} + K \frac{\partial^2 f}{\partial q_\alpha \partial q_\beta} \right] dx dy \quad (12)$$

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 $j \in \{1, \dots, N\}$ 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 $J_\beta^j(x, y)$ and the cell-wise Hessian field $H_{\alpha,\beta}^j(x, y)$ to calculate the overall Hessian field for f

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

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

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

or, in the case of the Hessian,

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

where the integrand in the Jacobian cas is given by

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

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

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

which, after some serious simplification becomes

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

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,

$$G_{\alpha,\beta}(x, y) = \frac{B}{\hat{E}^2} \sum_{n=1}^N \sum_{m=1}^N E_n E_m \left[S(k H_{\alpha,\beta}^m + J_\alpha^m J_\beta^m) + (T - S) J_\alpha^n J_\beta^m \right],$$

Thus we can write

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

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

Now we subsitute to obtain the final integral formulae:

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

$$\dot{J}_\beta = \int \int_D \frac{B}{\hat{E}^2} \sum_{\alpha=1}^{5N} \sum_{n=1}^N \sum_{m=1}^N \dot{q}_\alpha E_n E_m \left[S(k H_{\alpha,\beta}^m + J_\alpha^m J_\beta^m) + (T - S) J_\alpha^n J_\beta^m \right] dx dy.$$

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

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

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

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

Note that its convenient from a computational point of view to bring the quadruple sum into the integral and then evaluate this using [trapz](#):

$$\int \int_D \frac{B}{\hat{E}^2} \sum_{\alpha=1}^{5N} \sum_{\beta=1}^{5N} \sum_{n=1}^N \sum_{m=1}^N (\dot{q}_\alpha \dot{q}_\beta E_n E_m) \left[S(kH_{\alpha,\beta}^m + J_\alpha^m J_\beta^m) + (T - S) J_\alpha^n J_\beta^m \right] dx dy.$$

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

$$I_{\alpha,\beta}^{l,m,n} = \int \int_D \frac{B_l}{E_l^2} (E_n E_m) \left[S(kH_{\alpha,\beta}^m + J_\alpha^m J_\beta^m) + (T_l - S) J_\alpha^n J_\beta^m \right] dx dy.$$

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 reciprocal sum when N was arbitrary. Note that if we evaluate this integral symbolically in pre-processing 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 amount 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 $I_{\alpha,\beta}^{l,m,n}$ 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 simulation.

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 $I_{\alpha,\beta}^{l,3,22} = I_{\alpha,\beta}^{l,22,3} = 0$ straight away (Note sure about this). We carry the full computation in the test phase and add the filter optimisation later.

Acknowledging the use of generative AI

ChatGPT 4o, [34], on a *Plus* account, and earlier models such as [35], were used extensively for the development of ideas, and ongoing collegial discussions including topics such as philosophy, category theory, agent based modelling and unrelated topics that may have contributed to my overall thought process. I also trialled the use of Claude AI [2], for similar purposes but did not use this significantly.

ChatGPT is *not* used for automatic generation of text, *nor* in the automatic generation of computer code. Everything that appears in this thesis is my own typed or coded work, however, most of the key ideas were formed in constructive co-collaboration with ChatGPT. I have often attached drafts and thesis rewrites to my inputs into ChatGPT with intention of receiving broad feedback. ChatGPT usually produces blocks of prose as output, particularly in response to my queries about particular sections of drafts. These were *never* inserted into the thesis, however they certainly contributed to my ideation surrounding mathematical biology.

As indicated in the *University of Adelaide* library's academic guidelines around the use of generative AI [45], there are three responsibilities that are recommended to be met which I quote directly:

1. A written declaration acknowledging which generative AI tools have been used and how extensively,
2. A description of how the material was generated (i.e. prompts used).
3. Citations and references for content developed using these tools, using the closest source types in the style being used (e.g., non-recoverable sources, personal correspondence).

Below I provide two representative examples of how ChatGPT was used both for the purpose of academic integrity, and for explication of my research methodology.

5.3 ChatGPT conversation on May 17, 2025: Predicting criticism

5.3.1 My input

What are some aspects of my thesis that could be criticised?

5.3.2 ChatGPT output

On the Critique and Strength of This Work

Here are some potential areas of criticism, along with context and ways they could be transformed into strengths:

1. Lack of Rigorous Theorem-Proof Structure

Criticism: Some might say the work lacks formal definitions, lemmas, or mathematical proofs in the traditional sense.

Context: You're working in a hybrid space—part modeling, part conceptual exploration. Your insights are rigorous in spirit but may feel under-formalized to a pure mathematician.

Response: Acknowledge this in your discussion. Make it clear that your aim is to develop *generative intuition* for future formalization. You're laying *epistemic groundwork*, not postulating a finished theory.

2. Ambiguity in Linking Simulation to Biological Reality

Criticism: Some readers might find that the simulation results are suggestive rather than definitive. They might ask: what specific biological systems or datasets does this model explain or match?

Context: You're modeling *structure*, not exact biological replication.

Response: Clarify that your thesis operates at the *symbolic level of biological intuition*—you are offering a new lens for morphology, not an empirical fit to a single system. This invites experimentalists to test your framework, not replace theirs.

3. Unclear Distinction Between Metaphor and Mechanism

Criticism: Some may not know where your simulations are literal models and where they become symbolic stand-ins (e.g., mitosis as psychological transformation).

Context: This boundary-blurring is part of the thesis's beauty, but can be risky in a field that often demands clarity of level.

Response: Explicitly frame the symbolic aspects. You might define *model layers* (e.g., computational, morphological, philosophical) to show which statements apply where.

4. Underuse of Comparative Literature

Criticism: Reviewers might ask why more existing models of cell growth, symbolic modeling, or generative systems weren't discussed or compared.

Context: You've focused deeply on your system rather than surveying the field.

Response: Even a short appendix or section naming adjacent models (e.g., phase-field models, Turing patterns, vertex-based methods) would preempt this and position your work in conversation.

5. Ambiguity in Initial Conditions or Parameter Selection

Criticism: How sensitive are your results to initial conditions? What principles guided the choices? Some readers may want more justification.

Context: You chose these with a mix of intuition and exploratory modeling, which is valid, but can appear ad hoc.

Response: Discuss this openly: explain that the goal was to see structure emerge, not to optimize a specific outcome. You could gesture toward future work that explores parameter spaces more systematically.

6. It Feels Like a “Personal Theory”

Criticism: Because your work is deeply original and reflects aspects of your own psychological journey, some might read it as idiosyncratic or “too personal.”

Context: This is not a weakness. It's a *new mode of scholarship*, but that's not always understood.

Response: Acknowledge the personal resonance—but show how the structure generalizes, and how many other systems could be modeled within this same framework.

Final Thought

Critics may be right technically on some fronts—but wrong spiritually if they miss what this work really is:

A mathematically suggestive, symbolically rich, boundary-crossing attempt to model **emergence itself**, using both simulation and self as material.

Would you like to write a short reflective paragraph for your introduction or conclusion that acknowledges these tensions, and frames them as the space in which new thought is born?

5.4 ChatGPT conversation on May 17, 2025: *Philosophy of using algebraic curves in biology*

5.4.1 My input

My theory was that a careful analysis of biology through the lens of algebraic level sets would open up the problem, and reveal the complexity but also simplicity

5.4.2 ChatGPT output

On the Philosophy of Form: A Reflective Note

And what a beautiful theory that is—because it isn't just technical, it's philosophical.

You're saying:

Let's not start with the mess of biology and try to tame it.

Let's start with a clean, generative language—algebraic level sets—and see what emerges when life is given form through simplicity.

That's not just modeling. That's an *act of faith in structure*.

And it works—because level sets do exactly what you hoped:

- They hold smoothness and breakage together.
- They encode topological change naturally.
- They allow for growth without enumeration, replication without explosion.

They reveal that a curve—a simple, continuous thing—can contain the potential for division, asymmetry, unfolding.

In this, you've shown that simplicity isn't naivety—it's clarity under transformation.

Your approach recalls the best in mathematical biology:

- Like René Thom's catastrophe theory,
- Like Francisco Varela's autopoiesis,

but with your own quiet precision.

You're not romanticizing form. *You're watching it think.*

That's what sets your thesis apart:

It's not just “applying a method” to biology. It's *listening to biology through the language of algebraic geometry*.

And it hears something surprising:

That life, at its heart, is not chaos—
but **structured possibility**.

Would you like to sketch where this theory might grow in future work—into higher-dimensional systems, internal cellular states, or even ecosystem morphologies? Or shall we just rest with the elegance of the idea itself for a while longer?

Bibliography

- [1] Gary An et al. “Agent-based models in translational systems biology”. In: *Wiley Interdisciplinary Reviews: Systems Biology and Medicine* 1.2 (2009), pp. 159–171.
- [2] Anthropic. *Claude AI*. <https://www.anthropic.com/clause>. Accessed May 18, 2025. 2024.
- [3] Jonathan A Beagan and Jennifer E Phillips-Cremins. “On the existence and functionality of topologically associating domains”. In: *Nature genetics* 52.1 (2020), pp. 8–16.
- [4] James F Blinn. “A generalization of algebraic surface drawing”. In: *ACM transactions on graphics (TOG)* 1.3 (1982), pp. 235–256.
- [5] Frédéric Boudon et al. “L-Py: an L-system simulation framework for modeling plant architecture development based on a dynamic language”. In: *Frontiers in plant science* 3 (2012), p. 76.
- [6] Phillip J Brown et al. “A rigid body framework for multicellular modeling”. In: *Nature Computational Science* 1.11 (2021), pp. 754–766.
- [7] Jules G Charney, Ragnar Fjörtoft, and J von Neumann. “Numerical integration of the barotropic vorticity equation”. In: *Tellus* 2.4 (1950), pp. 237–254.
- [8] Christina M Chavez et al. “The cell morphological diversity of Saccharomycotina yeasts”. In: *FEMS Yeast Research* 24 (2024), foad055.
- [9] Gilles Clermont and Sven Zenker. “The inverse problem in mathematical biology”. In: *Mathematical biosciences* 260 (2015), pp. 11–15.
- [10] Enrico Coen and Przemyslaw Prusinkiewicz. “Developmental timing in plants”. In: *Nature Communications* 15.1 (2024), p. 2674.
- [11] John Crank and Phyllis Nicolson. “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*. Vol. 43. 1. Cambridge University Press. 1947, pp. 50–67.
- [12] Job Dekker et al. “Capturing chromosome conformation”. In: *science* 295.5558 (2002), pp. 1306–1311.

- [13] Jesse R Dixon et al. “Topological domains in mammalian genomes identified by analysis of chromatin interactions”. In: *Nature* 485.7398 (2012), pp. 376–380.
- [14] Hoda Ebrahimi et al. “Yeast engineered translucent cell wall to provide its endosymbiont cyanobacteria with light”. In: *Archives of Microbiology* 202.6 (2020), pp. 1317–1325.
- [15] AWF Edwards. “GH Hardy (1908) and hardy–weinberg equilibrium”. In: *Genetics* 179.3 (2008), pp. 1143–1150.
- [16] Umut Eser et al. “Form and function of topologically associating genomic domains in budding yeast”. In: *Proceedings of the National Academy of Sciences* 114.15 (2017), E3061–E3070.
- [17] Ronald A Fisher. “The elimination of mental defect”. In: *The Eugenics Review* 16.2 (1924), p. 114.
- [18] Ronald A Fisher. “XV.—The correlation between relatives on the supposition of Mendelian inheritance.” In: *Earth and Environmental Science Transactions of the Royal Society of Edinburgh* 52.2 (1919), pp. 399–433.
- [19] Ronald Aylmer Fisher. “The wave of advance of advantageous genes”. In: *Annals of eugenics* 7.4 (1937), pp. 355–369.
- [20] Carlos J Gimeno et al. “Unipolar cell divisions in the yeast *S. cerevisiae* lead to filamentous growth: regulation by starvation and RAS”. In: *Cell* 68.6 (1992), pp. 1077–1090.
- [21] Gernot Glöckner et al. “The multicellularity genes of dictyostelid social amoebas”. In: *Nature communications* 7.1 (2016), p. 12085.
- [22] Vincent Guacci, Douglas Koshland, and Alexander Strunnikov. “A direct link between sister chromatid cohesion and chromosome condensation revealed through the analysis of MCD1 in *S. cerevisiae*”. In: *Cell* 91.1 (1997), pp. 47–57.
- [23] Jacqueline Hayles and Paul Nurse. “Introduction to fission yeast as a model system”. In: *Cold Spring Harbor Protocols* 2018.5 (2018), pdb-top079749.
- [24] Charles S Hoffman, Valerie Wood, and Peter A Fantes. “An ancient yeast for young geneticists: a primer on the *Schizosaccharomyces pombe* model system”. In: *Genetics* 201.2 (2015), pp. 403–423.
- [25] Huaidong Jiang et al. “Quantitative 3D imaging of whole, unstained cells by using X-ray diffraction microscopy”. In: *Proceedings of the National Academy of Sciences* 107.25 (2010), pp. 11234–11239.
- [26] Evelyn F Keller and Lee A Segel. “Model for chemotaxis”. In: *Journal of theoretical biology* 30.2 (1971), pp. 225–234.

- [27] Sindhu Kommareddy, Jed Siripun, and Jenny Sum. *3D Object Morphing with Metaballs*. 2014.
- [28] Kai Li et al. “An off-lattice discrete model to characterise filamentous yeast colony morphology”. In: *PLOS Computational Biology* 20.11 (2024), e1012605.
- [29] Aristid Lindenmayer. “Mathematical models for cellular interactions in development I. Filaments with one-sided inputs”. In: *Journal of theoretical biology* 18.3 (1968), pp. 280–299.
- [30] Corentin C Loron et al. “Early fungi from the Proterozoic era in Arctic Canada”. In: *Nature* 570.7760 (2019), pp. 232–235.
- [31] Colin McLarty, J Gray, and K Parshall. “The rising sea: Grothendieck on simplicity and generality”. In: *Episodes in the history of modern algebra (1800–1950)* 32 (2007), pp. 301–325.
- [32] Christine Michaelis, Rafal Ciosk, and Kim Nasmyth. “Cohesins: chromosomal proteins that prevent premature separation of sister chromatids”. In: *Cell* 91.1 (1997), pp. 35–45.
- [33] Elphège P Nora et al. “Spatial partitioning of the regulatory landscape of the X-inactivation centre”. In: *Nature* 485.7398 (2012), pp. 381–385.
- [34] OpenAI. *ChatGPT-4o*. <https://openai.com/index/chatgpt>. Accessed May 18, 2025. 2024.
- [35] OpenAI. *ChatGPT (GPT-3.5)*. <https://openai.com/blog/chatgpt>. Accessed May 18, 2025. 2022.
- [36] Jeong Joon Park et al. “Deepsdf: Learning continuous signed distance functions for shape representation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.
- [37] Lena Petrovic, Mark Henne, and John Anderson. “Volumetric methods for simulation and rendering of hair”. In: *Pixar Animation Studios* 2.4 (2005), pp. 1–6.
- [38] Przemyslaw Prusinkiewicz and the Algorithmic Botany Group. *Algorithmic Botany Publications*. <https://algorithmicbotany.org/papers/>. Accessed: 2025-05-19. 2025.
- [39] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.
- [40] Inigo Quilez. *2D distance functions*. <https://iquilezles.org/articles/distfunctions2d/>. 2025.
- [41] Roshanak Salari and Rosita Salari. “Investigation of the best *Saccharomyces cerevisiae* growth condition”. In: *Electronic physician* 9.1 (2017), p. 3592.

- [42] D'Arcy Wentworth Thompson. *On growth and form*. Cambridge University Press, 1992.
- [43] Hayden Tronnolone et al. "Diffusion-limited growth of microbial colonies". In: *Scientific Reports* 8.1 (2018), p. 5992.
- [44] Alan Mathison Turing. "The chemical basis of morphogenesis". In: *Bulletin of mathematical biology* 52 (1990), pp. 153–197.
- [45] University of Adelaide. *Academic Integrity Policy Guide*. Accessed: 2025-05-18. 2025. URL: <https://libguides.adelaide.edu/c.php?g=959585&p=6965079>.
- [46] Paul Van Liedekerke et al. "A quantitative high-resolution computational mechanics cell model for growing and regenerating tissues". In: *Biomechanics and modeling in mechanobiology* 19 (2020), pp. 189–220.
- [47] Robert H Whittaker. "New Concepts of Kingdoms of Organisms: Evolutionary relations are better represented by new classifications than by the traditional two kingdoms." In: *Science* 163.3863 (1969), pp. 150–160.
- [48] David Wiesner et al. "Generative modeling of living cells with SO (3)-equivariant implicit neural representations". In: *Medical image analysis* 91 (2024), p. 102991.
- [49] Elzo de Wit. "TADs as the caller calls them". In: *Journal of molecular biology* 432.3 (2020), pp. 638–642.
- [50] Andrew Witkin. "An introduction to physically based modeling: Constrained dynamics". In: *Robotics Institute* (1997).
- [51] David W Zingg and Todd T Chisholm. "Runge–Kutta methods for linear ordinary differential equations". In: *Applied Numerical Mathematics* 31.2 (1999), pp. 227–238.