Estimating Parameter Distributions for Biophysical Simulation of Monocultural Microbial Populations

by

Jonathan Chalaturnyk

A report
presented to the University of Waterloo
in fulfillment of the
research requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

Stochastic agent based models of monoculture bacterial colonies are an important first step in modelling more complex systems. We present a basic growth simulation of monoculture bacterial colonies using the cell simulation software BSim. These simulations are highly parameterized and require careful calibration to ensure model outputs are true to experimental observations. A set of features extracted from raw simulation data are proposed to be utilized in the calibration of BSim simulations. These features underwent a sensitivity analysis with respect to the growth simulation parameters. Several features are identified as promising candidates for utility in calibration. We then perform an Approximate Bayesian Computation calibration of a simple growth simulation in BSim.

Acknowledgements

I would like to thank Prof. Brian Ingalls for their continued support and guidance over the course of my Master's degree. This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca) and Compute/Calcul Canada.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent history, the field of synthetic biology has developed into an exciting research domain that combines engineering, biology and mathematical principles, all of which have been heavily studied (Benner and Sismour, 2005; Yeh and Lim, 2007; Cameron et al., 2014). During the field's infancy, the concept of genetic circuits was introduced as an exciting way to induce and control gene expression, specifically in *E. coli* microbial systems (Gardner et al., 2000). This novel framework allowed for more complex circuits to be designed and tested, which has led to new discoveries in metabolic engineering and microbiology (Stephanopoulos, 2012). Additional applications of synthetic biology appear in agriculture (Roell and Zurbriggen, 2020), material sciences (Tang et al., 2021), immunology (Sepich-Poore et al., 2021) and therapeutics (Charbonneau et al., 2020).

As expected with modelling complex, dynamical systems, there is a strong desire for accurate and efficient computational frameworks. Computing complex, biological systems *in silico* allows for the rapid prototyping and testing of experiments over a design space that would be otherwise unfeasible *in vitro*. A modelling approach for biological systems known as agent based modelling (ABM) has gained significant popularity recently, especially for synthetic biology applications (Gorochowski, 2016; An et al., 2017). ABM approaches attempt to model the individual behaviour of distinct biological entities within their environment, which allows for the careful control over agent to agent interactions and global dynamics of the system's population.

Studying microbial systems of practical interest often involve multiple, distinct populations of bacteria interacting with one another. Therefore, it is crucial that we accurately model the growth dynamics and morphology of monoculuture (single progenitor) bacterial populations *in silico*. Once accomplished, ABM models can be created from baseline

growth behaviours that are highly characterized and more interesting bacterial properties can be added from specific *in vitro* experimental conditions. For example, a case study performed by Matyjaszkiewicz et al. (2017) utilizing a stochastic cell simulation software called BSim, showed a successful *in silico* implementation of an oscillating, synthetic microbial system designed first observed *in vitro* by Chen et al. (2015). BSim is particularly successful due to its ability to hybridize the model approach by characterizing the spatio-temporal nature of a bacterial colony with individual agent dynamics and underlying chemical signalling with ordinary or delay differential equations. There are many other useful computational tools to perform biophysical simulations of monocultural, microbial populations such as gro, CellModeller, and BacSim; see Pinheiro and Gorochowski (2016) for a more comprehensive list.

Stochastic modelling approaches, such as BSim (Matyjaszkiewicz et al., 2017), require careful calibration of model parameters to ensure that simulation outputs mimic desired target data (i.e. experimental data). In particular, we are concerned about calibration methods that take into account microscopy data, where the data captures bacterial colony growth and morphology, as well as genotypic expression that may be encoded in a genetic circuit and expressed through fluorescence. Calibration and validation of stochastic ABM models in finance and economics is very common (Fievet and Sornette, 2018; Fagiolo et al., 2019; Pietzsch et al., 2020), however, there is a scarce amount of literature regarding calibration methods for the spatio-temporal ABM simulations we are interested in.

A major roadblock for calibrating stochastic ABM simulations is the inability to obtain a computationally tractable likelihood function, due to the implausibly large task of producing all possible simulation trajectories. To overcome this problem, likelihood free methods such as Approximate Bayesian Computing (ABC) are an increasingly popular and effective choice (Sunnåker et al., 2013; Liepe et al., 2014; Lintusaari et al., 2017). ABC methods use a reduced dimension representation of simulated and target data to determine the similarity between the two, with an iterative refinement of the similarity criteria. In essence, the goal of ABC is to estimate a posterior distribution for the parameters used to generate simulations, such that the distance (similarity) between simulated and target data is minimized. It is critical that appropriate representations of the data are calculated such that very little information loss occurs. Often a set of summary statistics or features are collected from continuous stochastic simulations with relative ease and have successfully been utilized in ABC calibration experiments (Hartig et al., 2011; Fearnhead and Prangle, 2012b). However, for stochastic ABM models and in particular spatio-temporal models of cell growth, there is no standardized method for obtaining suitable features to be applied in ABC methods. A number of studies (You et al., 2018; Dell'Arciprete et al., 2018; Doumic et al., 2020) suggest emergent features that could be extracted from typical

bacterial simulation or microscopy images.

In this study, we utilize the stochastic cell simulation software BSim (Matyjaszkiewicz et al., 2017) to generate *in silico* monoculuture populations of *E. coli*. Our goal is to accurately capture single colony dynamics using BSim, which requires the proper modelling and parameterization of cell behaviour for these simulations. In Section 2 we outline the specific details for our BSim simulations and the underlying parameterization and biophysical assumptions. If cell to cell interactions are properly implemented to reflect experimentally observed *E. coli*, the stochastic BSim simulations should be capable of producing results sufficiently similar to real experimental data. Sufficiency is a loose criteria set by specific use cases, however, our expectations are that the BSim models can be appropriately calibrated against a target experimental data set. Section 3 outlines a set of data features motivated by (You et al., 2018; Dell'Arciprete et al., 2018; Doumic et al., 2020) to be utilized during calibration. In the following Section (4), we investigate the sensitivity of the proposed data features by performing a one-at-a-time (OAT) sensitivity analysis on our parameters outlined in Section 2. For our BSim simulations of monocluture, bacterial growth, calibration involves the estimation of weighted parameter distributions for later use in generating validated model output. Finally, we present a case study in Section 5, where we attempt to infer parameter distributions from synthetic data for a simple growth simulation in BSim.

Chapter 2

BSim Cell Simulator

2.1 Overview

To simulate bacterial populations, the simulation software BSim was utilized to accomplish this task (Matyjaszkiewicz et al., 2017). BSim uses an agent-based modelling (ABM) approach for simulating cell dynamics, where individual agents represent unique entities in a biological environment. The major benefit of ABM comes from the fact that a developer has full control over the mathematical rules that govern the behaviour and interactions of each individual agent, or in this case, bacteria. This fact is very attractive in synthetic and systems biology as it suggests an inherent ability to quickly implement and test experimental designs *in silico*, that would otherwise take long periods of time *in vivo*. Additionally, BSim incorporates stochastic effects into the modelling structure of cell simulations, with the hopes of capturing the stochastic qualities of natural systems. In the simplest case of modelling cell growth, stochastic effects appear by randomly sampling distributions to select the individual cell growth rate and the position of daughter cells after its progenitor divides.

2.2 Bacterial Monoculture Modelling in BSim

It has been shown that many interesting applications arise from investigating microbial systems where multiple colonies interact (González-Pastor et al., 2016). Therefore, it is critical that the morphology of individual (monoculture) cell colonies are accurately modelled before exploring multi-colony interaction. For this reason, modelling monoculture

growth, from a single cell initial condition, was used as the baseline experiment for this study. To simplify the analysis further, we look at a two-dimensional model of cell growth, restricting bacteria from growing into the z plane. Consequently, the cell simulation outputs resemble that of two-dimensional images taken from typical microscopy experiments.

To model the two-dimensional system, a canonical growth simulation was crafted that employed 8 adjustable parameters. These parameters modify simulation behaviour that both directly and indirectly impact cell interaction, growth and division. Table 2.1 shows the parameter names in BSim, along with their numerical ranges. The numerical ranges were determined throughout the numerical experiments performed in this study. More detailed explanations of each parameter are provided in the relevant subsections.

Other important characteristics of the BSim simulations are the individual agents' attributes at every time step. Firstly, since we are modelling rod shaped *E. coli* bacteria, each cell is represented by two coupled points denoted by x_1 and x_2, with some fixed radius r defining a capsular shape for all cells. We denote the position of the center of the cell with $\bar{x}$. BSim also keeps track of other pertinent quantities such as cell length (L) and orientation (ϕ). All simulations are initialized with a single cell placed at the center of a 1800 units by 1800 units square with a total simulation time of 5 hours and a $dt = 0.03$ hours.

To consult the exact modelling implementations, please refer to https://github.com/ingallslab/bsim/blob/master/run/BasicSimulation2D/BasicSimulation2D.java.

Parameter Description	**BSim Name**	**Symbol**	**Low**	**High**
Initial Asymmetrical Growth	`init_growth_asym`	α_i	0	1
Scale Asymmetrical Growth	`asymmetry_scale`	α_s	0.5	1
Twist Cell at Birth	`birth_twist`	β	0	0.5
Sticking Force Constant	`k_stick`	k_s	0	1
Overlap Force Constant	`k_overlap`	k_o	10	500
Internal Force Constant	`k_int`	k_i	0	5
Sticking Force Range	`contact_rng`	δ_r	0	0.5
Sticking Force Contact Threshold	`contact_damping`	δ_d	0	4

Table 2.1: Full list of BSim parameters used to adjust growth simulation behaviour. A short description accompanies the naming convention used in BSim, as well as an abbreviated symbol used throughout the report.

2.3 BSim Parameterization

2.3.1 Spring Forces

BSim utilizes a suite of spring forces to describe several different forces imposed by cells on each other. These spring forces are generically described by,

$$F_{\text{spring}} \propto k_{\text{spring}}(\tilde{L}_{\text{spring}} - \tilde{L}^{\text{rest}}_{\text{spring}})^p \tag{2.1}$$

where F_{spring} is proportional to the difference between spring length $\tilde{L}_{\text{spring}}$ and the resting spring length $\tilde{L}^{\text{rest}}_{\text{spring}}$, and scaled by the corresponding spring constant k_{spring}. For a value of $p = 1$ this expression is Hooke's law for a linearly elastic spring force, and for $p \neq 1$ we have a non linear spring force. The three spring forces that are relevant to this study utilize different values of p and will be outlined next.

The first of the three spring forces is the internal force, which is parameterized via the spring constant k_i. At a time step t, the simulation will propose a growth amount ΔL that updates the length of the cell to $L_{\text{new}} = L_{\text{old}} + \Delta L$. In the internal spring force calculation, the resting spring length $L^{\text{rest}}_{\text{spring}} = 0$ and $L_{\text{spring}} = ||x_2 - x_1|| - L_{\text{new}}$, where x_2 and x_1 are the positions of the cell nodes. It's important to note that the node to node distance $||x_2 - x_1||$ will almost always be less than L_{new} because the positions are not updated to reflect the proposed growth step until after all the forces are resolved. Therefore, the spring force $F_{\text{internal}} \propto k_i(||x_2 - x_1|| - L_{\text{new}})^2$ acts outwardly on each node to ensure that no other forces prevent the cell from realizing it's new length L_{new}. A simple depiction of this force is presented in Figure 2.1. In the rare case that $||x_2 - x_1|| > L_{\text{new}}$ this force would be able to pull nodes back to the desired length.

A high valued k_i will ensure that any forces acting along the growth axis of the cell, do not effect the new length L_{new} at the current time step t. If a small k_i is used, this allows neighbouring forces to impact the new length of the cell after the growth amount is proposed. In practice, a small value of k_i will cause cell lengths to increase very slowly (due to compression from other forces), with very few reaching their threshold length in an appropriate time scale.

Figure 2.1: Schematic of internal force acting on cell nodes in BSim simulations.

Next we consider the overlapping force between cells, which ensures that two neighbouring cells do not physically occupy the same volume. The strength of this force is controlled with the spring constant k_o. As shown in Figure 2.2, if two cells' positions overlap, the spring force will act such that they are pushed apart. Two cells i and j are considered overlapping if their neighbour distance $d_n = ||\bar{x}_i - \bar{x}_j||$ is less than $2r$, where r is the radius of our rod shaped cells. The resting spring length $L^{\text{rest}}_{\text{spring}} = 0$ and the overlap force is described by $F_{\text{overlap}} \propto k_0(2r - d_n)^2.5$. A small k_o allows cells to coalesce because the force won't be strong enough to continually push cells past overlapping, whereas a large value will guarantee separation occurs.

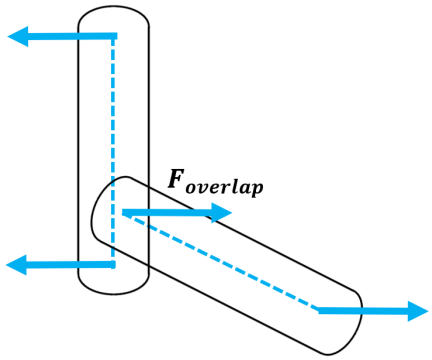

Figure 2.2: Schematic of overlap force acting on neighbouring cells in BSim simulations.

Finally, the third spring force is the sticking force. In reality, *E. coli* possess surface organelles called pili (Paranchych and Frost, 1988) that allow for an individual bacteria to adhere to other bacteria. Modelling these filament structures directly would be too complex and instead the adhesion effects are modelled with a sticking spring force. Figure 2.3 shows a simple depiction of how the sticking force is applied to two cells. The strength of this force is again parameterized with the corresponding spring constant k_s.

The specific implementation of the sticking force takes into account the pairwise distances between all nodes of two neighbouring cells. For each of the 4 pairwise distances, we model a spring with either a short resting spring length $L^{\text{rest}}_{\text{short}}$ or a long resting spring length $L^{\text{rest}}_{\text{long}}$. The short and long axes correspond to the shortest node distance between neighbour cells and the longer (diagonal) distance respectively. We denote the shorter distance as d_{short} and long distance as d_{long}. Therefore, along the two short axes we apply a sticking force $F_{\text{sticking,short}} \propto k_s(d_{\text{short}} - L^{\text{rest}}_{\text{short}})$ and along the two long axes we apply a sticking force $F_{\text{sticking,long}} \propto k_s(d_{\text{long}} - L^{\text{rest}}_{\text{long}})$.

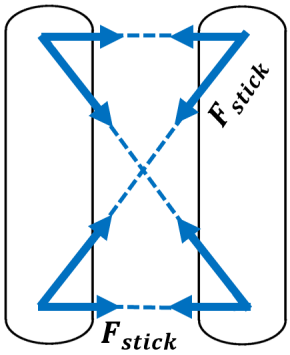

Figure 2.3: Schematic of sticking force acting on neighbouring cells in BSim simulations.

2.3.2 Sticking Force Limits

In Figure 2.3, we see the force diagram of how two cells are drawn towards each other when the sticking force is applied. If two cells are within reach of each others pili, which is defined as a threshold distance `contact_rng` (δ_r), the sticking force model will attract the two cells to each other. We also define a dampening effect, such that the sticking force stops acting on neighbouring cells when they have sufficient contact. Numerically, the amount of contact is calculated by drawing two bounding boxes around neighbouring cells and finding the perimeter of the overlapping section between the two boxes. The parameter `contact_damping` (δ_d) is the threshold value of this perimeter at which the sticking force turns off between neighbouring cells. An example of two intersecting bounding boxes between neighbouring cells in shown in Figure 2.4.

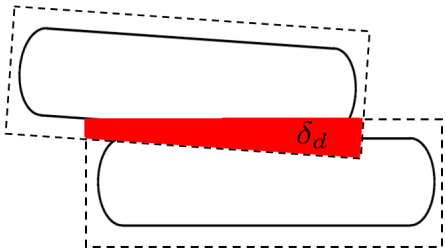

Figure 2.4: Schematic of contact damping parameter δ_d effect in BSim. Note that the perimeter of the red intersection area is compared to δ_d to control the strength of sticking force.

2.3.3 Asymmetric Growth

When observing *E. coli* growth, it has been shown that the two poles of a rod shaped bacteria may grow at differing rates (Kysela et al., 2013). Therefore, during growth of that bacteria, the forces exerted by each pole of the bacteria may differ and consequently alter the morphology of an entire colony over time. We choose to model and tune this behaviour using two parameters, `init_growth_asym` (α_i) and `asymmetry_scale` (α_s).

At a given time step t during the simulation, a certain growth amount ΔL is proposed to extend the length of the cell. For symmetric growth, this length extension is applied equally to both ends of a cell. To account for asymmetric growth, we define an initial degree of asymmetry α_i bound between 0 and 1. This parameter scales the amount of internal force felt by each node x_1 and x_2. It is implemented in BSim such that x_1 feels a greater internal force, which ensures this end of the cell grows by ΔL at time t and the node x_2 will grow by some amount less than ΔL.

The second parameter used to model asymmetrical growth is the scaling parameter α_s. This has a relatively simple effect of tapering off the initial asymmetrical growth α_i, as the cell grows. Once the cell reaches a length of $L = \alpha_s L_{\text{thresh}}$, the cell will start to grow symmetrically until it reaches its predefined division threshold length L_{thresh}. If we denote the internal force felt during symmetric growth as f_{internal} and the internal force felt during asymmetric growth as F_{internal}, we have the following equations that describe the dampening model as,

$$F_{\text{internal}}(x) = f_{\text{internal}}(x)(1+\omega) \quad , \text{ at } x = x_1 \tag{2.2}$$
$$F_{\text{internal}}(x) = f_{\text{internal}}(x)(1-\omega) \quad , \text{ at } x = x_2 \tag{2.3}$$

where,

$$\omega = \max\left(0 \ , \ \alpha_{\text{i}}\left(1 - \frac{\text{L}}{\alpha_{\text{s}}\text{L}_{\text{thresh}}}\right)\right). \tag{2.4}$$

2.3.4 Cell Division

Once a cell reaches a length $L = L_{\text{thresh}}$, this is the length at which the cell will divide. In BSim, the program simply creates two new cells at this time step, with the parent's nodal information passed onto the daughter cells. Additionally, BSim adds a small perturbation vector $\beta\bar{v}$ to one of the daughter cells orientations' to ensure they are not perfectly aligned.

Here, $\bar{v} \in \mathbb{R}^2$ is a random uniform vector scaled by the `birth_twist` parameter β. The new position of the daughter cell right after birth is given by $\bar{x}_{\text{birth}}$. To perturb the position right after birth, we calculate

$$\bar{x}_{\text{new}} = \bar{x}_{\text{birth}} + \beta\bar{v} \tag{2.5}$$

where $\bar{x}_{\text{new}}$ is the new perturbed position of the daughter cell. Figure 2.5 shows a depiction of how this random orientation might look in the simulation.

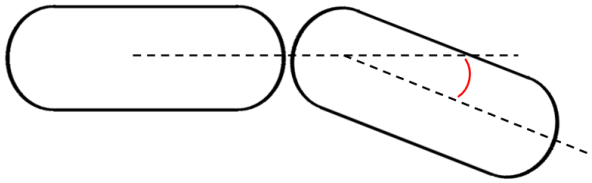

Figure 2.5: Schematic of two daughter cells after a division event in BSim, where one cell has their position perturbed by $\beta\bar{v}$.

Chapter 3

Feature Definitions

3.1 Overview

In order to compare experimental data to simulated data sets of *E.coli* growth, we search for some means of numerical comparison. Since BSim, and many other biophysical models are stochastic (Goel and Richter-Dyn, 2016), a one-to-one comparison of experiment and simulation is impossible and therefore more careful measures must by employed. In general, the comparison of raw, stochastic data is performed by seeking a set of summary statistics (Fearnhead and Prangle, 2012b) that are reduced dimension representations of the data. In this study we refer to summary statistics as *features* of the data. Many of the features are calculated with image analysis tools and therefore require two dimensional images of cell colonies. Although *E. coli* are not perfectly capsular in shape, it is easiest to approximate their shape as a capsule with fixed radius and a variable cell length. Therefore, using our data containing cell position, length and orientation, we can generate an image $\mathbf{X}$ of size N_{pixels} of the cell colony using the Python package *Pillow* (Clark, 2015), with a total of N_{cell} objects represented as a capsule. Features that do not require images of the cell colonies are instead calculated from the raw data.

3.2 Scalar Features

There are a set of four features obtained by calculating a single quantity from the data, either because the feature involves collapsing all data points into one quantity or because it is related to a single time point in the dynamics of the *E. coli* colony. One of these four

features is calculated for every cell at a given time t, where the distribution of values is collapsed into an averaged value over the total cell population at every time step. This approach is a notably naive approach for obtaining the trend over time, however, was chosen for simplicity. These features are motivated by (Doumic et al., 2020) and are adapted to fit our modelling framework.

3.2.1 Density

The first of the scalar features is the density of the single *E. coli* cell colony. This is an easy feature to calculate at any time point and is done through simple image analysis. For a given image at time t, we leverage the Python packages *opencv* (Bradski, 2000) and *skimage* (Van der Walt et al., 2014) to output pertinent measurements. First, we obtain the total area A of the colony to be used in the density calculation. This is obtained by generating a contour around the edges of the colony and setting all pixels to 255 within the contour and counting the total number of pixels equal to 255. Then, to obtain the density of the colony given an image $\mathbf{X}$, where pixels corresponding to cells are set to an integer p_{cell} we get

$$f_{\text{density}} = \Big(\sum_{i=1}^{N_{\text{pixels}}} \mathcal{I}(\mathbf{X}_i = p_{\text{cell}}) \Big) / A \tag{3.1}$$

where, f_{density} corresponds to the density feature and $\mathcal{I}$ is an indicator function. The two image creation steps mentioned in this formulation are shown in Figure 3.1.

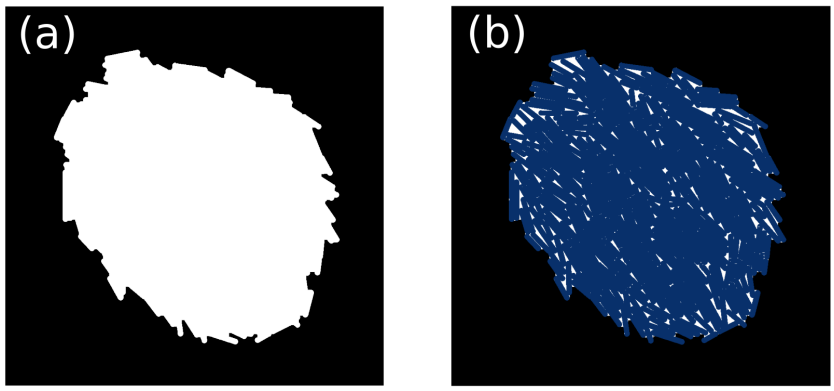

Figure 3.1: (a) Sample image of post processed cell colony with approximated edge contour and filled in pixels. (b) Overlayed image of raw cell colony data, on top of filled in colony.

3.2.2 Dyad Structure

As noted in (Doumic et al., 2020), rod shaped bacteria, and in particular *E. coli*, form a consistent pattern in their colony morphology when the colony reaches a total population of four bacteria. Specifically, they form a tetrad formation shown in Figure 3.2.

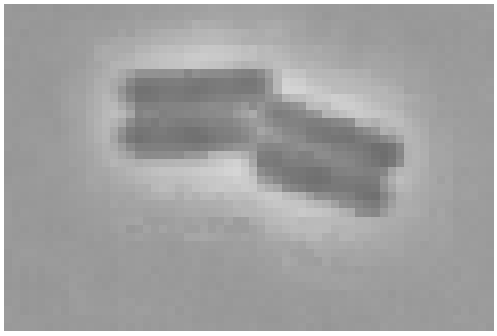

Figure 3.2: Phase microscopy image of a four cell *E. coli* colony forming an aligned, tetrad structure.

With the tetrad formation observed at a high frequency in *E. coli* growth, capturing this effect in simulation is paramount and therefore requires a unique feature used in calibration. We utilize a similar quantity defined in (Doumic et al., 2020) that takes into

account the orientation of each bacteria in the two cell colony, which we refer to as the dyad structure. In experimental data, the two cell colony trends towards having two cells stacked on top of each other (see either left or right pair in Figure 3.2). Therefore, we aim to capture this by calculating,

$$f_{\text{dyad}} = \frac{(\bar{x}_1 - \bar{x}_2) \cdot \bar{x}_2}{||\bar{x}_1 - \bar{x}_2||\ ||\bar{x}_2||} \tag{3.2}$$

where $\bar{x}_1$ and $\bar{x}_2$ are the positions of the center of each of the bacteria in the two cell colony. Notice that if the two bacteria are parallel, f_{dyad} will equal 1 and subsequently will equal 0 if they are orthogonal.

3.2.3 Aspect Ratio

The third scalar feature that is obtained is the aspect ratio of a single *E. coli* colony. The aspect ratio feature was shown to change for different parameterized adhesion strengths in simulations performed by (Doumic et al., 2020) and therefore presents an interesting measure of cell colony difference between unique simulation configurations. Aspect ratio is collected by fitting an approximate ellipse to the enclosed colony region shown in Figure 3.1. An ellipse inherently has a major axis a and minor axis b that defines it's shape and eccentricity. The two axes are are then combined to calculate the aspect ratio. The equation for the aspect ratio of an ellipse and hence the cell colony, is given by the ratio,

$$f_{\text{ap}} = \frac{a}{b} \tag{3.3}$$

where f_{ap} corresponds to the feature for aspect ratio. A depiction of the ellipse fitted to a sample cell colony is presented in Figure 3.3.

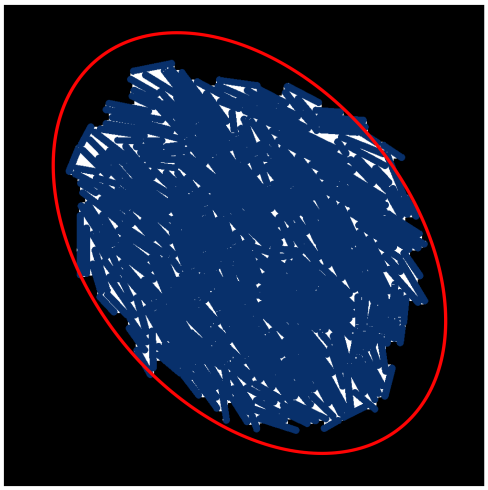

Figure 3.3: Sample image of post processed cell colony with approximate ellipse fitted to the colony area.

3.2.4 Anisotropy

Individual cell orientation, and the orientation of neighbouring cells can have dramatic effects on the morphology evolution of bacterial colonies. Therefore, it is critical to capture a measure of how cells are oriented with respect to their neighbours over the duration of colony growth. We utilize the measure of local anisotropy as described by (Doumic et al., 2020), which captures the degree to which neighbouring cells are similarly oriented to to one another.

The calculation of anisotropy is as follows. For every cell i in a colony of size N, the average projection matrix $\boldsymbol{P}_i$ is calculated with,

$$\boldsymbol{P}_i = \frac{1}{N_{\text{neighbour}}} \sum_{j=1}^{N} \begin{pmatrix} \cos\phi_j^2 & \cos\phi_j \sin\phi_j \\ \cos\phi_j \sin\phi_j & \sin\phi_j^2 \end{pmatrix} \mathcal{I}(d(i,j) < d^*) \tag{3.4}$$

where $d(i,j)$ is the center to center distance between two cells given by,

$$d(i,j) = ||\bar{x}_i - \bar{x}_j||. \tag{3.5}$$

The threshold distance d^* for cells to be considered neighbours, is set at a fixed value of $60\mu m$ and $N_{\text{neighbour}}$ is the number of cells that satisfy $d(i,j) < d^*$ for a given cell i. The

local anisotropy λ_i of each cell i is then calculated with,

$$\lambda_i = \lambda_{\max}(\boldsymbol{P}_i) \tag{3.6}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\boldsymbol{P}_i$. Since we end up with a local anisotropy value for each cell, at every time step in data, we collapse the collection of values at each time step to an average anisotropy value given by,

$$f_{\text{anis}} = \frac{1}{N}\sum_{i=1}^{N}\lambda_i \tag{3.7}$$

where f_{anis} is the feature corresponding the "total" anisotropy of a colony. With this method of calculating anisotropy within the cell colony, it is important to note that a λ_i value equal to 1 indicates that all neighbouring cells within $d(i,j) < d^*$ are oriented in the same direction and a λ_i value equal to 0.5 suggests the opposite scenario. The two extremes can be seen in Figure 3.4.

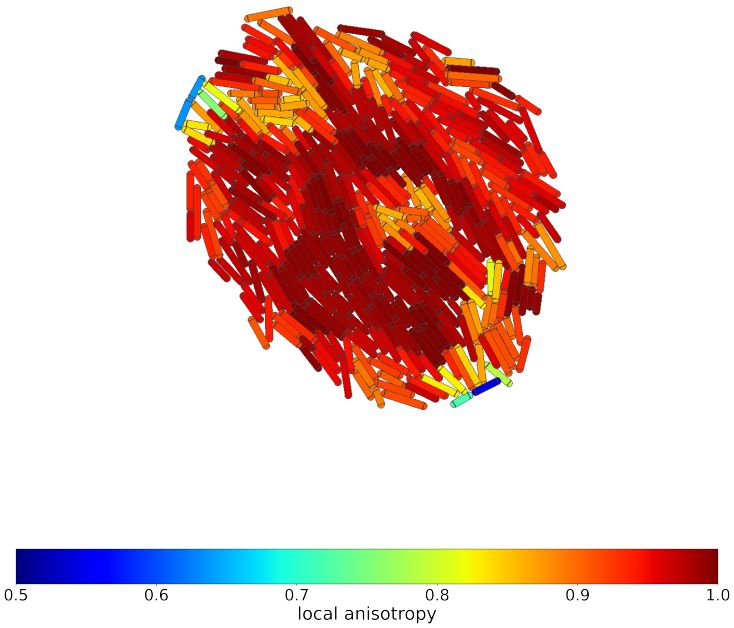

Figure 3.4: Local anisotropy values for individual cells in a sample BSim simulation output. Anisotropy values range from 0.5 (not oriented) to 1.0 (highly oriented).

3.3 Distributed Features

The first four features were of a classification, where at a given time point in the data, we can extract a single measure for each cell in the total population (or for a single time point in the case of dyad structure). The final three features differ from the scalar features for two reasons. First, they are collected as a distribution of a quantity over the total population of a single colony and second, they are only calculated at the final time step of the data. Collecting the distributed features at the end of a simulation, or in experimental data, gives more statistics and therefore a greater chance of obtaining a precise measurement.

3.3.1 Cell Orientation on Colony Boundary

As suggested by Dell'Arciprete et al. (2018), there is a tendency for *E. coli* at a large colony size to align themselves with the boundary of the colony, which can be seen in Supplementary Figure A.1. This suggests that a simulation which correctly characterizes the distribution of angles for cells on the boundary of a colony could act as a very powerful comparison tool between simulated and real data. In Dell'Arciprete et al. (2018) the authors represent cell positions as a continuum, whereas we have discrete positions $\bar{x}$ for each individual cell object in the data. Therefore, a modified approach is taken to obtain a distribution of cell orientations on a colony's boundary.

The cell positions at the final time step are collected and make up a set of points V. An algorithm called *alphashape* (Asaeedi et al.; Kirkpatrick and Seidel, 1983) is applied to the set of points V, where a subset of points $U \subset V$ are obtained. In practice, the coordinates $U \subset V$ are used to define a bounding polygon whose vertices are given by the points in U. The discrete set of points U chosen by the alphashape will later be used as the foundation for a continuous boundary representation. Each point in U corresponds to a cell that lies exactly on the boundary.

The alphashape algorithm takes in a single parameter $\alpha \in \mathbb{R}$, which defines the radius of $\frac{1}{\alpha}$ of a generalized disk. A connected component of the bounding alpha shape polygon is drawn between two points v_1 and v_2 in V, if a generalized disk of radius $\frac{1}{\alpha}$ intersects v_1 and v_2 but contains no other points in the set V. For a small α, we have a large generalized disk, which invokes the requirement that two points be on the outside of a cluster of points, i.e. cluster of points representing a cell colony. In fact, if we set $\alpha = 0$, the generalized disk becomes a half plane and the alpha shape algorithm reduces down to computing the convex hull. All computations of alpha shapes in this study utilize $\alpha = 0.01$. The points

that make up all of the connected components define $U \subset V$ and we then have a set of coordinates that can be used to define a boundary for a single cell colony.

We then fit a cubic spline to the points in U for a continuous representation of the boundary. A sample image of the splines generated for $\alpha = 0$ (convex hull) and $\alpha = 0.01$ is given in Supplementary Figure A.2. With a spline fitted to the alpha shape boundary of a cell colony, we can then easily compute the cell's orientation corresponding to points in U, with respect to the cell boundary and obtain an overall distribution for our data. Supplementary Figure A.3 showcases a sample result from the orientation calculation for a simulated cell colony. Distributions of cell orientation are then generated by making a histogram of the cell orientations, with bins of 1° width. All cell angles are transformed to be between 0 and 90° to avoid degeneracy of angles for rod shaped bacteria. We denote the feature relating to the distribution of cell orientation on the boundary of a colony as f_{angle}. An example distribution is shown in Figure 3.5.

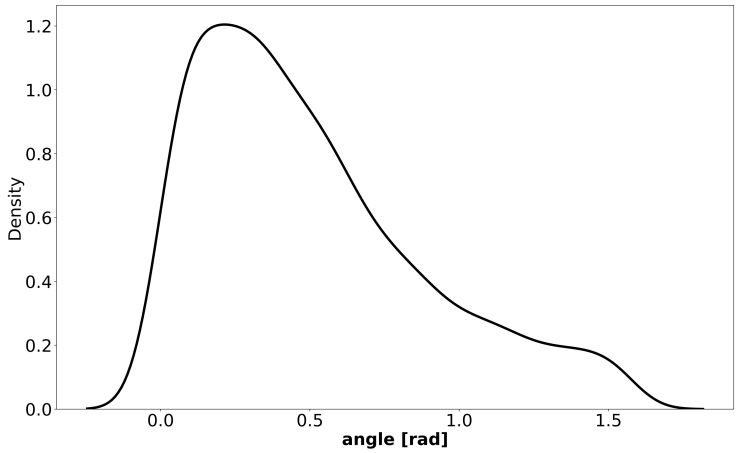

Figure 3.5: Distribution of cell orientations with respect to cell colony boundary for a sample BSim simulation ensemble of 500 runs.

3.3.2 Oriented Patch Size

In a study performed by You et al. (2018), they investigated the existence of oriented microdomains within a rod-shaped bacterial cell colony. A microdomain, or patch, is defined as a cluster of neighbouring cells which are similarly oriented within a tolerance angle θ_t. Clusters of neighbouring cells are also dependent on a length scale l that defines the distance at which cells are considered neighbours. It was found by You et al. (2018)

that the size of a patch, which corresponds to the total number of cells in that patch, depends on certain growth specific factors, such as growth rate and division length. This suggests that probing the number of cells in oriented patches for a fixed l and θ_t could yield different patch size frequency for colonies of differing growth or simulation conditions.

We utilized the method by You et al. (2018) to cluster cells in our data into patches of similar orientation for a length scale of $l = 25\mu m$ and $\theta_t = 1°$. The result of applying this method to BSim simulation data is shown in Figure 3.6. It is clear that regions of cells that exhibit very similar or identical orientation are coloured the same, indicating they belong to the same oriented patch. Taking data, like the data we see in Figure 3.6, we can collect a distribution corresponding to the frequency of patch size in that data set or ensemble. An example of a resulting distribution is shown in Figure 3.7. We denote the feature related to the distribution seen in Figure 3.7 as f_{patch}.

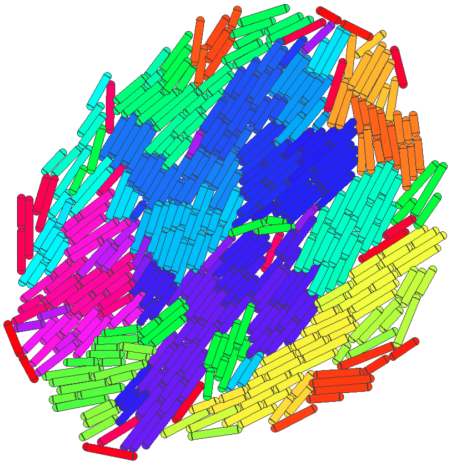

Figure 3.6: Sample result for patch size calculation. Neighbouring cells that are coloured the belong to the same patch.

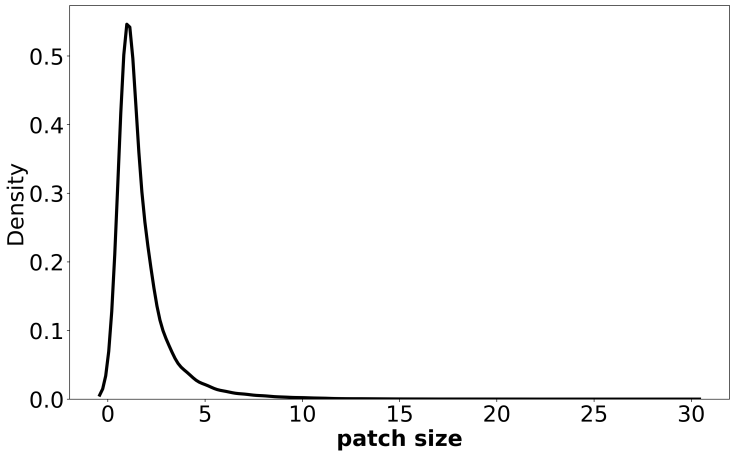

Figure 3.7: Distribution of patch size for a sample BSim simulation ensemble of 500 runs.

3.3.3 Cell Age and Distance within a Colony

To track an individual cell's age, an addition to BSim was made in order to ensure the lineage is calculated correctly. In the biological context, the cell's age corresponds to a daughter cell's node inheriting the same nodal material as it's parent cell. Put simply, each daughter cell will have half of the grandmother cell and one half of "new" material. Therefore, the bacteria in BSim carry an age associated to each node x_1 and x_2, where each node age is incremented each time that node becomes part of a daughter cell after division. The entire cell's age is then defined as the maximum age over the two node ages. Intuitively, one would then expect the oldest cells to be pushed to the exterior of a cell colony, since they contain the material from the first, single cell. Because of this, we then group all of the cells of equal age and collect the relative positions of those cells with respect to the colony's centroid. An overall trend in these distributions should favour younger cells being more centralized and older cells being near the exterior of a colony. Figure 3.8 shows a sample simulated colony with the observed behaviour.

Since each age group has their own distribution of cell distances, we collapse each of those distributions down into a mean value μ_{age} that is then used to create a point (age,μ_{age}), for each of the observed ages in a simulation ensemble or data set. A set of points (age,μ_{age}) for age $= 1, ..., \max(\text{age})$ is denoted as the feature f_{age}. These points allow us to plot curves which show how the mean changes for age group, as well as how the overall trend changes when model parameters are adjusted. An example curve created from BSim simulation data is shown in Figure 3.9.

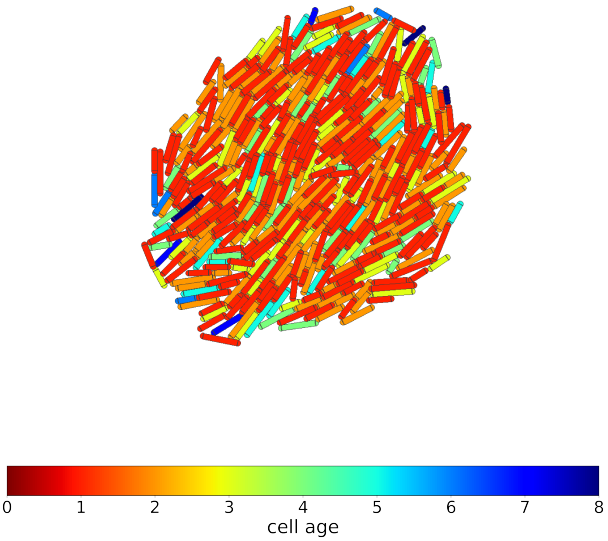

Figure 3.8: Sample BSim simulation with cell colours corresponding to their age. As expected, older cells in blue trend towards the exterior of the colony.

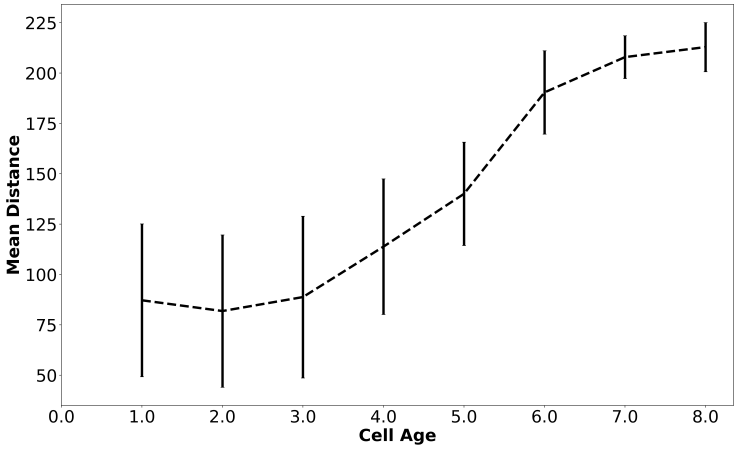

Figure 3.9: Sample curve corresponding to mean distance of cell's to their colony centroid as a function of cell age. Error bars are the standard deviation in cell distance for a particular age group. Statistics were collected from a BSim simulation ensemble of 500 runs.

Chapter 4

Feature Sensitivity Analysis

4.1 Methods

There are a total of seven features that are explored in this study, where each feature was determined to be either sensitive or non-sensitive to parameter changes in BSim. The main application of the sensitivity analysis is to determine whether the features defined in Section 3 could effectively measure a distance between simulated and experimental data. We are able to calculate a distance via the aforementioned features because they will be a transformation from individual agent data, to scalar or vectorized quantities that describe a global state of the agents.

The sensitivity analysis will not involve real data sets. However, the ability to compare simulations of different configurations in our parameter space should indicate the efficacy of our features for the task. Ideally, the set of seven features examined in this study should be able to capture differences between data sets when the seven BSim parameters are modified. If a feature is unable to capture differences, it's possible that additional experiments or modelling refinements could be added to address this problem in the future.

To determine what features were sensitive to parameter changes, a one-at-a-time (OAT) (Hamby, 1994) sensitivity analysis was performed for the BSim parameters. This involved performing a sweep between a single parameters high and low value, where we chose 25 steps between the extremes. Additionally, we defined a default value p_0 as the midpoint between the high and low value for each parameter. p_0 was treated as the comparison data point for the sensitivity analysis. Although parameter interdependence exists for the BSim modelling, an OAT sensitivity analysis was chosen for simplicity. For each parameter value, an ensemble of 500 simulations were ran using high performance computing

(HPC) resources courtesy of ComputeCanada. The simulation ensembles ensure that ample statistics are obtained for each of the data features and uncertainties are minimized from stochastic effects.

Unlike a formal sensitivity analysis, we did not use a quantitative measure of how sensitive a feature was to parameter perturbation and instead we relied on visualizations of each feature to inform that judgement. For the visualizations we employed three different methods. First, the features were compared across parameter values for only the final time point of the simulations in each ensemble. Then we extend the amount of data to comparing the feature trends over the full temporal evolution of a simulation. It is important to note that the temporal evolution of a cell colony is not tracked in time units, but by cell colony area, which in a solely growth simulation will increase monotonically over time. Finally, we calculated the distance between feature distributions for each parameter value compared to the relevant default parameter value p_0 using a variety of distance metrics. Distance metrics included the Wasserstein distance (Villani, 2009), Kullback-Liebler (KL) divergence (Kullback and Leibler, 1951) and the absolute integral difference. The Wasserstein and KL divergence metrics were calculated using the Python open source package *scipy* (Virtanen, 2020) and the integrals of distributions were calculated using numerical Simpson integration. An example of the three visualization types is shown in Figure 4.1.

Since we utilized visualization techniques to inform our understanding of each features sensitivity with respect to each parameter, it is important to guide the reader through an example of what is considered a sensitive feature. Referring to Figure 4.1.a there is a clear separation in the mean aspect ratio for each of the ensembles simulated for high, middle and low k_s values. Despite some overlap with the error bars we can confidently say that at the final time point of a simulation, different values of k_s yield different aspect ratio values. Extending the endpoint analysis to the full temporal evolution, as shown in Figure 4.1.b, we wish to see a separation in the feature over the full simulation. Again, we see that the aspect ratio evolution differs across multiple parameter values. Utilizing the final visualization shown in Figure 4.1.c requires that there is monotonic increase or nearly monotonic increase in integral distance between simulation ensembles. Specifically we wish to see deviation from p_0 for all other parameter values considered in their respective ranges. A strong deviation from p_0 suggests that a given data set with a "true" value corresponding to p_0 will yield features that are distinguishable to data sets with different parameter values. Each feature is said to be sensitive to a parameter by inspecting that features' visualization and looking for adequate separation between data of different simulated parameter configurations.

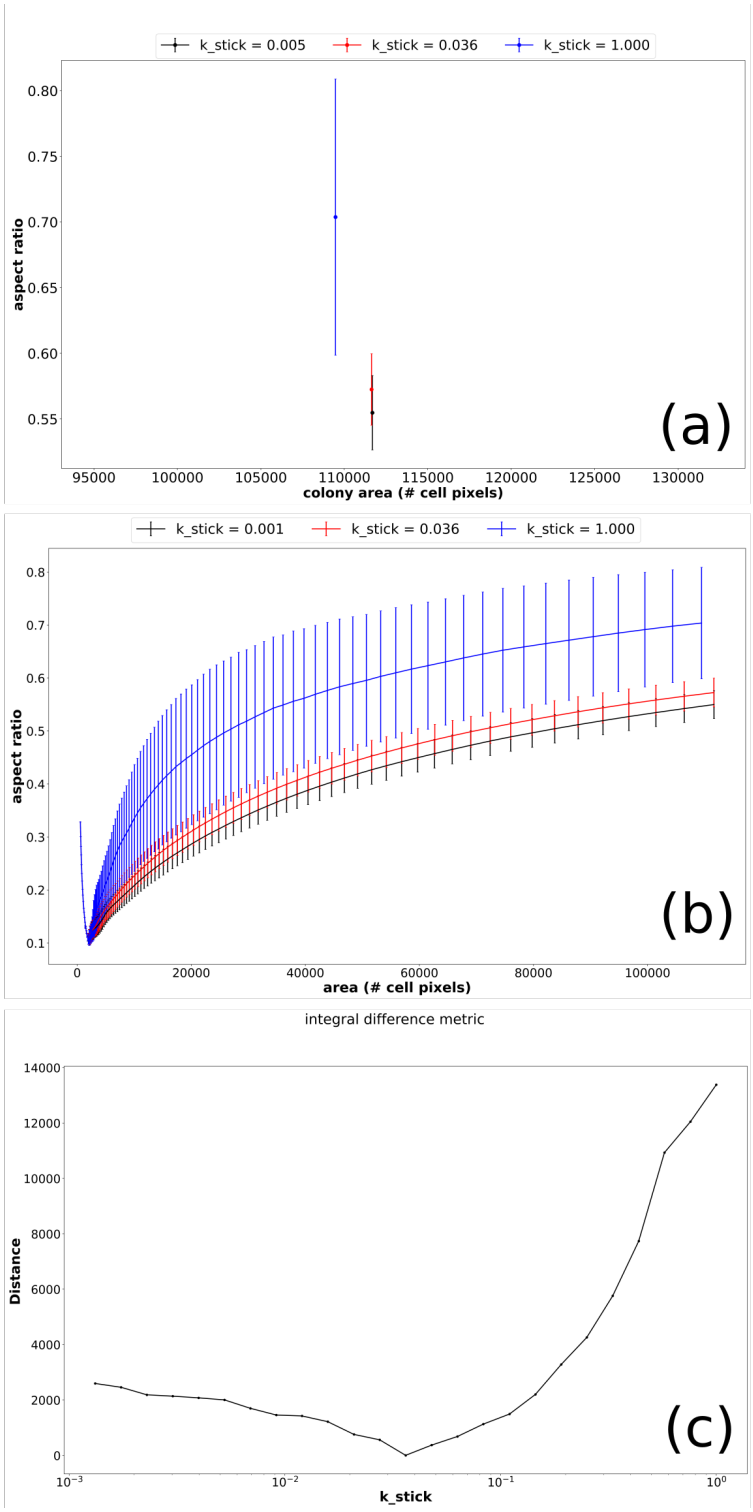

Figure 4.1: Example visualizations used in feature sensitivity analysis. (a) Endpoint comparison of aspect ratio feature f_{ap} between simulations of varying sticking force k_s. (b) Full temporal comparison, as a function of colony area, for f_{ap}. (c) Integral difference between curves in panel (b) corresponding to different k_s values. Integral difference is calculated between all parameter values and the central parameter value.

4.2 Results

4.2.1 Density

We tabulate the sensitivity results for the density feature f_{density} in Figure 4.2, where filled in cells corresponds to an observed sensitivity in f_{density} for the matching parameter in that row. It is clear that the three spring forces have an impact on colony density and this is expected, since the three spring forces k_s, k_i and k_o dictate the strength of forces exerted between neighbouring cells. As both k_i and k_o decrease, the density should increase since small internal and overlap forces would allow for cells to pack together more tightly. This effect can be observed in Supplementary Figures B.1 and B.2. The inverse effect is observed for k_s, where increasing the sticking force decreases the density, since "stickier" cells prevent cells from sliding past one another and therefore form a tightly packed colony. This is shown clearly in Figure 4.3. We also observed a sensitivity in f_{density} with respect to birth twist β, however, the effect is not as dramatic as the spring forces. For all other parameters, the density evolves nearly identically for each parameter values.

BSim Name	**Symbol**	f_{density}
`init_growth_asym`	α_i	
`asymmetry_scale`	α_s	
`birth_twist`	β	
`contact_rng`	δ_r	
`contact_damping`	δ_d	
`k_stick`	k_s	
`k_int`	k_i	
`k_overlap`	k_o	

Figure 4.2: Sensitivity results for density (f_{density}). Grey boxes correspond to sensitivity in f_{density} with respect to the BSim parameter in the same row.

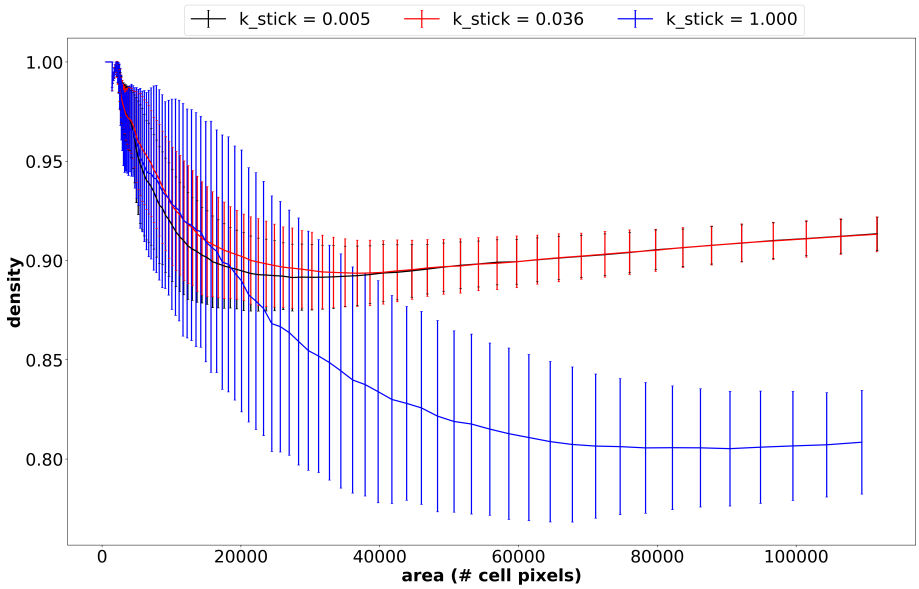

Figure 4.3: Mean density (f_{density}) as a function of cell colony area for three different sticking force constant k_s simulation ensembles. Error bars represent standard deviation of the density over the simulation ensemble.

4.2.2 Dyad Structure

For the dyad structure feature f_{dyad}, we observed promising results with respect to the birth twist parameter β only. We inspected the mean of f_{dyad}, defined in Equation 3.2, over simulation ensembles and plotted the mean with respect to the parameter value used to generate the ensemble. The resulting curve is shown in Figure 4.4 for ensembles collected from differing birth twist simulations. It is evident that there is an increase in the mean value as we increase β. This indicates that varying the degree to which the first daughter cell is rotated after division, will also determine how the parent and daughter cell are aligned at the next cell division event. By creating the same plot shown in Figure 4.4 for all other parameter ensembles, there was no noticeable change in the mean dyad structure. Ideally, the dyad structure would show more variation when changing the parameters for asymmetrical growth since one would expect that growth favouring one end of a cell would encourage two cells to slip past one another. It is possible that the modelling of asymmetrical growth may need to take into account more complicated effects.

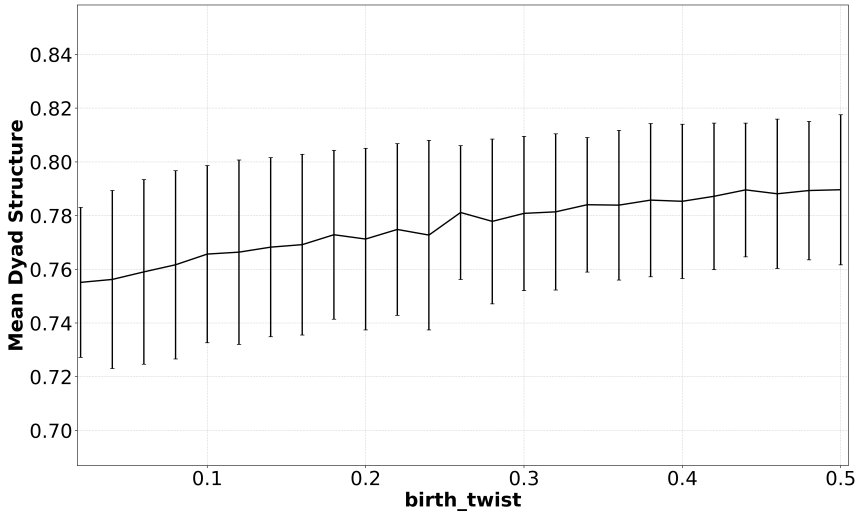

Figure 4.4: Mean dyad structure (f_{dyad}) for birth twist simulation ensembles. Birth twist values on x-axis refer to scale parameter β applied to random vector $\bar{v} \in \mathbb{R}^2$. A cell's position after birth is perturbed to $\bar{x}_{\text{new}} = \bar{x}_{\text{birth}} + \beta\bar{v}$, where $\bar{x}_{\text{birth}}$ is the position of the cell right after birth. Error bars represent the standard deviation of the dyad structure of simulation ensembles.

4.2.3 Aspect Ratio

The aspect ratio feature f_{ap} is closely related to cell colony shape and was therefore observed to be sensitive to many of the parameters considered in our BSim model. The results are tabulated in Figure 4.5. Both of the asymmetrical growth parameters α_i and α_s produce significantly different aspect ratios between their high and low parameter values. This is shown for α_i in Figure 4.6, where a similar result was found for α_s. The results in Figure 4.6 also depict the sensitivity trend observed for birth twist β and contact range δ_r, where there is a significant difference in aspect ratio as the colony grows in area. Of all of the parameters, k_s produced the most variation between it's high and low parameter values. The results for k_s are shown in Supplementary Figure B.3.

BSim Name	Symbol	f_{ap}
init_growth_asym	α_i	
asymmetry_scale	α_s	
birth_twist	β	
contact_rng	δ_r	
contact_damping	δ_d	
k_stick	k_s	
k_int	k_i	
k_overlap	k_o	

Figure 4.5: Sensitivity results for aspect ratio (f_{ap}). Grey boxes correspond to sensitivity in f_{ap} with respect to the BSim parameter in the same row.

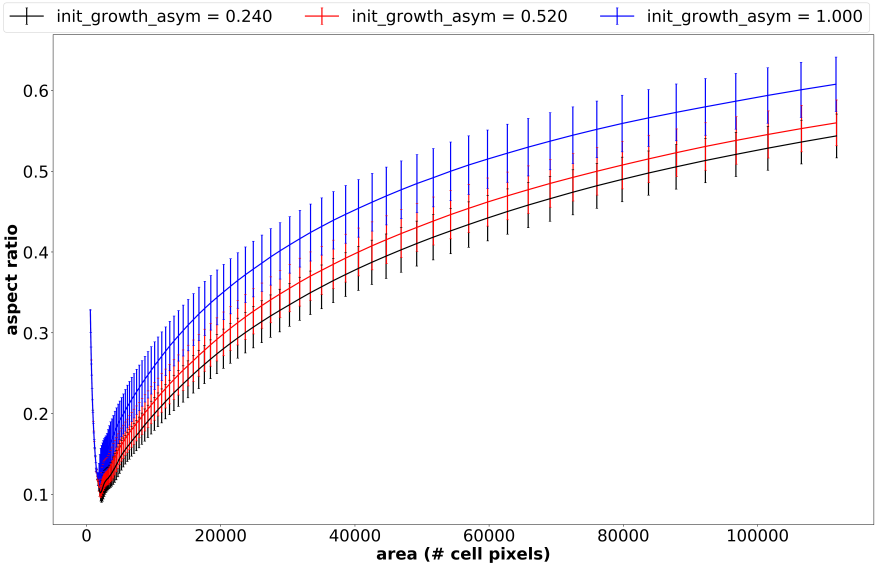

Figure 4.6: Mean aspect ratio (f_{ap}) as a function of cell colony area for three different initial asymmetrical growth α_i simulation ensembles. Error bars represent standard deviation of the aspect ratio over the simulation ensemble.

4.2.4 Anisotropy

Although the anisotropy feature f_{anis} is collapsed to a mean local anisotropy over all cells in a colony, it was still shown to be important for half of the parameters. The results for the f_{anis} sensitivity analysis are shown in Figure 4.7. A sample visualization for f_{anis} is shown for the sticking force parameter k_s in Figure 4.8. Here we see the most dramatic sensitivity result for f_{anis}, where changing the order of magnitude of k_s has a significant impact on the feature. This is intuitive for the sticking force since we expect low k_s values to produce highly compacted colonies, where neighbouring forces could easily align cells. For the other observed sensitivities in f_{anis} with respect to birth twist, internal force and overlap force constants, we saw significant overlap in the feature value over time for three different parameter values. However, we also noted that the separation between these curves was largest at their endpoints. This led to the conclusion that there may be suitable sensitivity in f_{anis} with respect to these three parameters, if considered only at the endpoint of the data. Further investigations should be performed to reduce the overlap in error between f_{anis} results. These investigations should also include an improvement on the calculation of the anisotropy feature.

BSim Name	**Symbol**	f_{anis}
`init_growth_asym`	α_i	
`asymmetry_scale`	α_s	
`birth_twist`	β	
`contact_rng`	δ_r	
`contact_damping`	δ_d	
`k_stick`	k_s	
`k_int`	k_i	
`k_overlap`	k_o	

Figure 4.7: Sensitivity results for mean local anisotropy (f_{anis}). Grey boxes correspond to sensitivity in f_{anis} with respect to the BSim parameter in the same row.

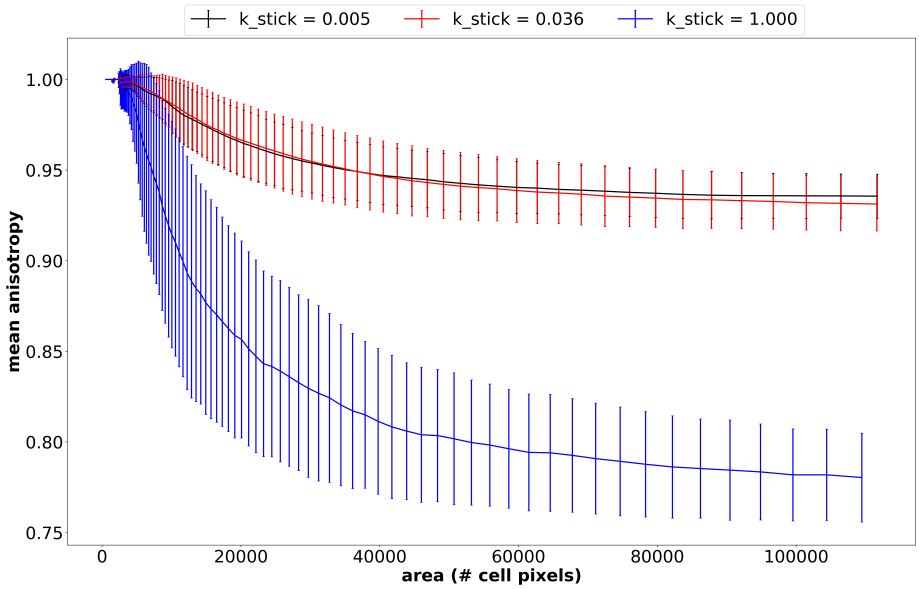

Figure 4.8: Mean anisotropy (f_{anis}) as a function of cell colony area for three different sticking force constant k_s simulation ensembles. Error bars represent standard deviation of the anisotropy over the simulation ensemble.

4.2.5 Cell Orientation on Colony Boundary

For the first of the distributed features f_{angle}, we observed a high impact by k_s and k_i. A collection of f_{angle} distributions are shown for various values in the parameter ranges for k_s and k_i in Figure 4.9. It is clear from these distributions that there is distinct variation of cell orientation on the boundary of a colony when changing the parameter values fir k_s and k_i. We also show this quantitatively by inspecting the absolute integral distance and KL divergence between distributions. These results are shown in Figure 4.10.

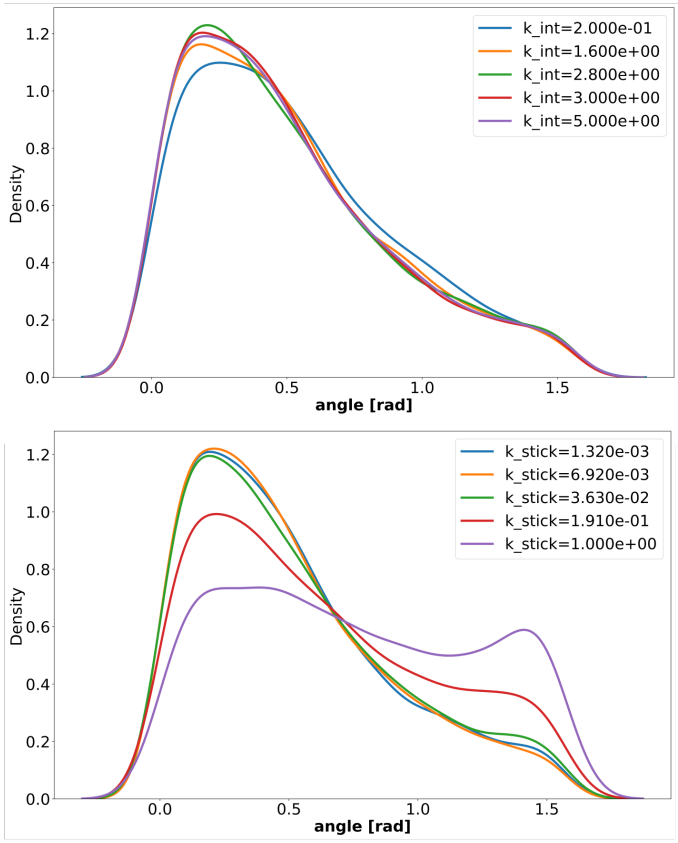

Figure 4.9: (a) Sample distributions for simulation ensembles with varying k_s. (b) Sample distributions for simulation ensembles with varying k_i. Distribution of cells on the boundary of cell colony and their orientation with respect to an alphashape, spline boundary. All data is collected at the final time point of the simulation.

It is important to note that the different metrics utilized to calculate distances between f_{angle} distributions yield much different results. Since the distributions for f_{angle} are of a similar shape for varying k_i, the only way to get a proper distance between distributions is by capturing the difference in peak height around 0.25 rad. Comparing the integral values of distributions for k_i was a valuable tool in capturing this effect. This result is clearly shown in Figure 4.10.b with increasing distribution difference for parameter values across the range of k_i values. Utilizing the KL divergence to compare distributions of f_{angle} for k_i proved to be unhelpful, which is shown in Figure 4.10.d. In contrast to k_i, we see that the KL divergence metric for comparing f_{angle} distributions for k_s values is very effective. It

is likely that differing distribution shapes in Figure 4.9 leads to better distance sensitivity seen in Figure 4.10.c.

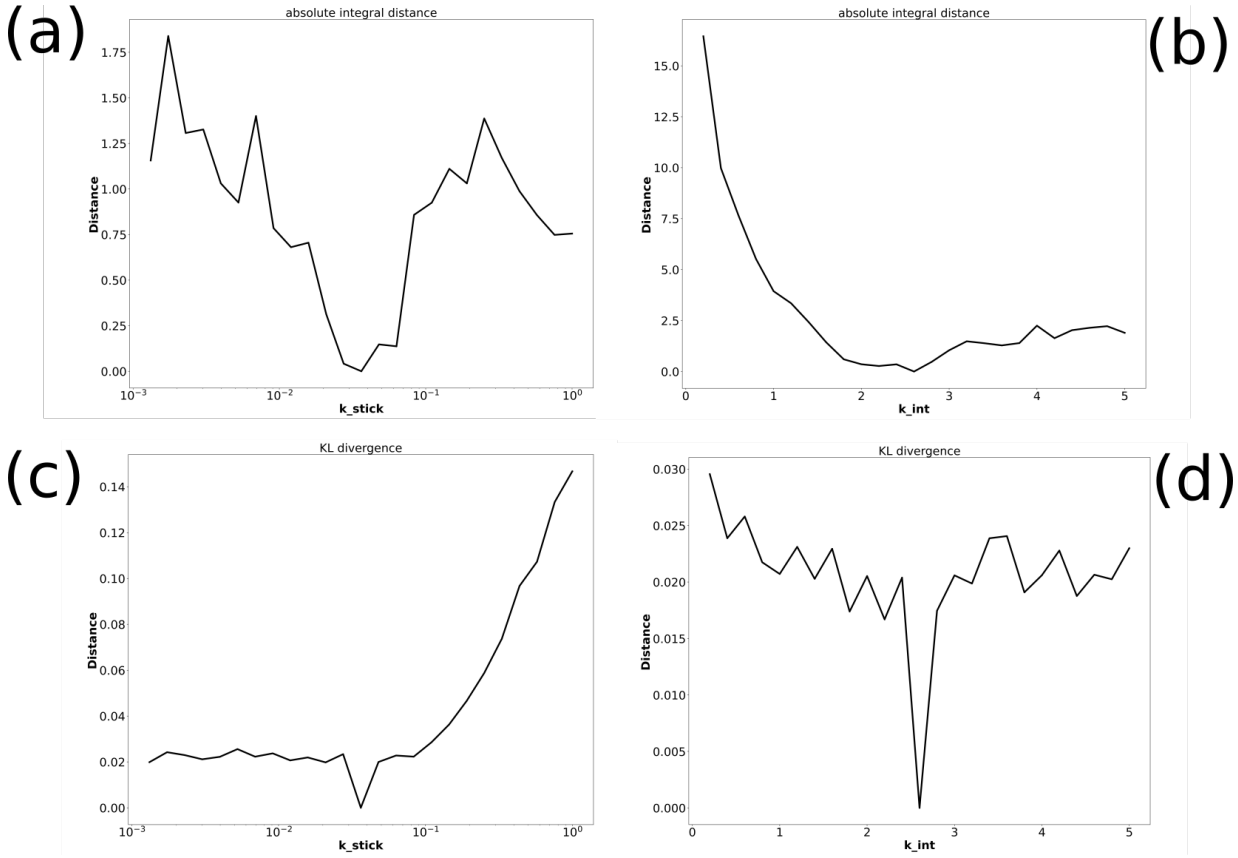

Figure 4.10: (a) Absolute integral difference between f_{angle} distributions for varying k_s. (b) Absolute integral difference between f_{angle} distributions for varying k_i. (c) KL divergence between f_{angle} distributions of varying k_s. (d) KL divergence between f_{angle} distributions of varying k_i. The central value in each of the parameter ranges are used as the comparison data set for each of the metrics.

4.2.6 Oriented Patch Size

For the feature f_{patch} it was found to be sensitive with respect to five of the eight parameters. These included the parameters β and δ_r for birth twist and contact range, as well as the three spring force constants k_s, k_i and k_o. To determine the sensitivity of f_{patch} we utilized

the distributions for oriented patch size as shown in Figure 3.7 and compared the distance between distributions for different simulation ensembles with the Wasserstein distance metric. Figure 4.11 shows the Wasserstein distance between patch size distributions of simulation ensembles corresponding to differing parameter values of k_o. There is a clear monotonic increase in the distance as we compare parameter values further from the central k_o value, indicating that there is sensitivity in f_{patch} with respect to k_o. We observed similar trends for the other two spring force constants k_s and k_i.

Figure 4.12 shows a much noisier Wasserstein distance calculation between simulation ensembles corresponding to δ_r. We also observed a similar trend for β simulation ensembles. Despite the presence of noise in the distance output, there is still a noticeable increase in distance as we deviate from the central parameter value, thus signifying the impact δ_r has on the f_{patch} feature.

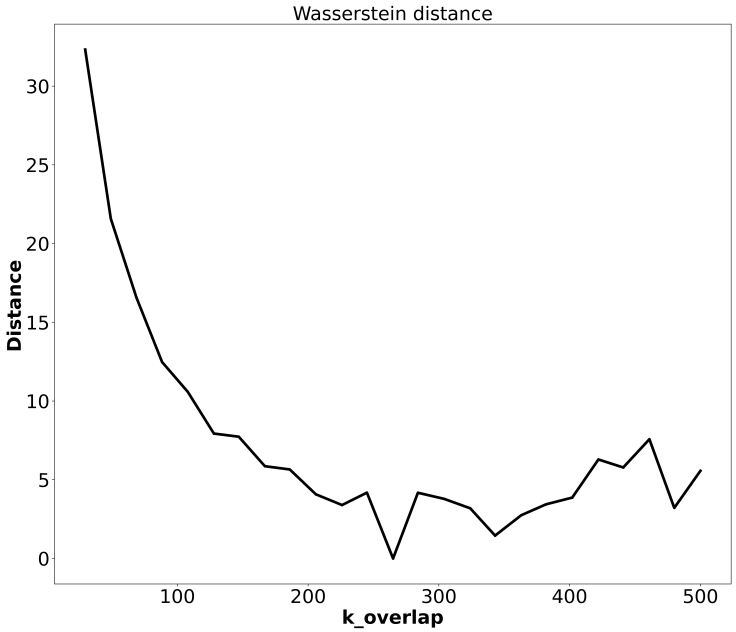

Figure 4.11: Wasserstein distance between f_{patch} distributions for varying overlap spring force constant k_o. All difference calculations were calculated with respect to the central value in the k_o range.

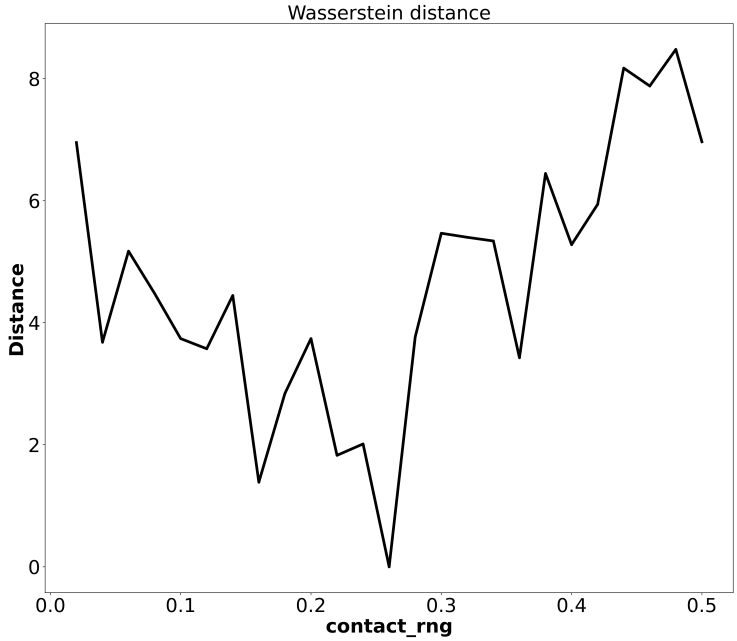

Figure 4.12: Wasserstein distance between f_{patch} distributions for varying contact range δ_r. All difference calculations were calculated with respect to the central value in the δ_r range.

4.2.7 Cell Age and Distance within a Colony

Figure 4.13 shows a sample comparison of f_{age} curves corresponding to mean distance of cells to their colony centroid, per age group in the cell colony. Each curve corresponds to a simulation ensemble simulated for a specific value of the asymmetric growth scaling parameter `asymmetry_scale`. We wished to observe curves, where mean distance varied across individual curves for a given age value. Effectively, an increasing separation between curves as cell age increases indicates that the feature f_{age} is able to capture the dynamics of certain age groups as we modify parameter values. Indeed as we modify the asymmetry scale parameter α_s, we see in Figure 4.13 that there is increased separation between f_{age} curves as α_s increases. Note that there are data points which extend out to a cell age of 10 for only three of the five curves in Figure 4.13, which is due to a small number of simulations in each ensemble producing cell colonies with cells of that age. Since we simulate stochastic simulations for a fixed time of 5 hours, we do not stop simulation for a particular age condition. We observed similar results for initial asymmetric growth α_i,

birth twist β, contact range δ_r and the three spring constants k_s, k_i and k_o. Each of the visualizations for these six other parameters follow a similar trend to that of Figure 4.13 with increasing separation between curves as the cell age increases. Therefore, we conclude that f_{age} is sensitive to the seven parameters mentioned above.

Determining the degree of sensitivity for the feature f_{age} was not a straightforward task. Since we collapse many distributions per simulation into an overall mean in the simulation ensemble, we are removing valuable information trapped within the original distance distributions per age group in a cell colony. We attempted an absolute integral difference comparison for curves shown in Figure 4.13 for the seven parameters that showed sensitivity in f_{age}, however the results showed a very noisy parameter versus distance relationship. An example of this noisy distance calculation is shown in Figure 4.14 for α_s simulation ensembles. Other distance metrics like Wasserstein distance and KL divergence were not calculated for these curves and should be explored further.

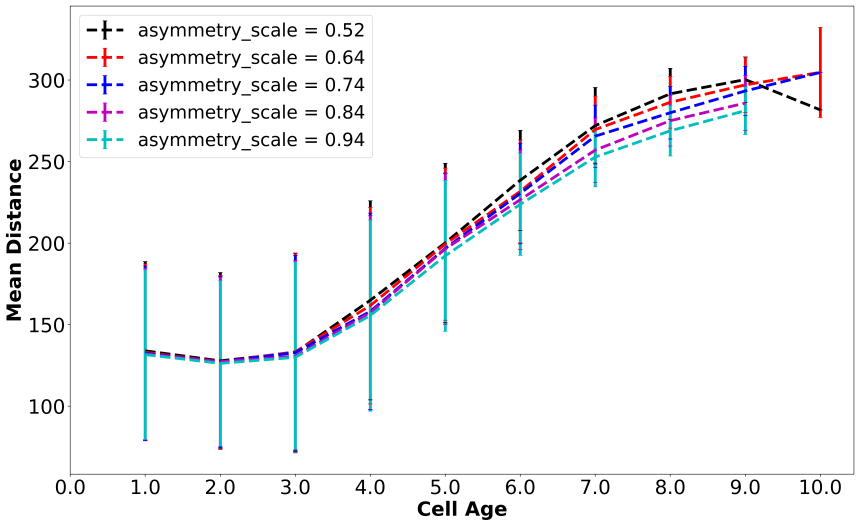

Figure 4.13: The mean cell distance to colony centroid within increasing age groups (f_{age}). Error bars represent the standard deviation in the distance with respect to the total distribution of distances over an entire simulation ensemble and age group.

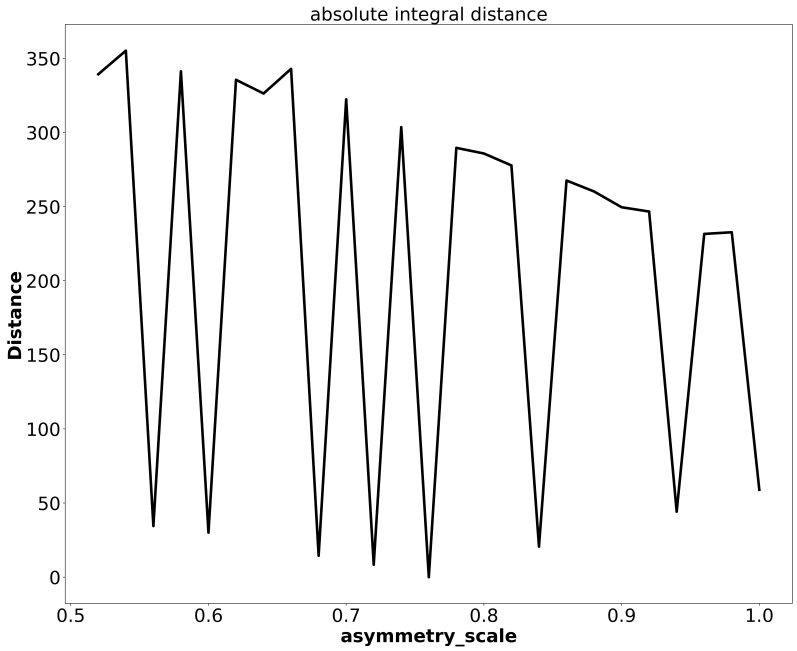

Figure 4.14: Absolute integral difference between f_{age} curves for varying asymmetry scale α_s simulation ensembles. All difference calculations were calculated with respect to the central value in the α_s parameter range.

4.3 Discussion

BSim Parameter	**Symbol**	f_{density}	f_{dyad}	f_{ap}	f_{anis}	f_{angle}	f_{patch}	f_{age}
`init_growth_asym`	α_i			■				■
`asymmetry_scale`	α_s			■				■
`birth_twist`	β	■	■	■	■		■	■
`contact_rng`	δ_r			■			■	■
`contact_damping`	δ_d							
`k_stick`	k_s	■		■	■	■	■	■
`k_int`	k_i	■			■	■	■	■
`k_overlap`	k_o	■			■		■	■

Figure 4.15: Full tabulation of sensitivity analysis results for all seven features with respect to the eight BSim growth simulation parameters. Grey boxes correspond to sensitivity of a feature column to the BSim parameter in the same row.

The final tabulation of sensitivity analysis results is provided in Figure 4.15. It is clear that some parameters had a greater effect on the features than others and this is likely due to a number of reasons. However, parameters which had little to no effect should not be treated as having no further utility. Notably, none of the features had a significant response with respect to the contact damping parameter δ_d. There is a possibility that our current growth simulations in BSim do not correctly model the desired behaviour for δ_d, which dampens the sticking force once ample contact between neighbouring cells occurs. More experimentation should also be performed after increasing the parameter range for δ_d, as its upper bound is dependent on the other parameter effects in our growth simulations. Specifically, the amount of contact between neighbouring cells is dependent on the contact range parameter δ_r. If certain values of δ_r increase the contact amount such that δ_d is too small of a threshold quantity, it will not perform its intended behaviour.

For the two asymmetrical growth parameters α_i and α_s it is important to note that only f_{ap} was observed to have any feedback with respect to parameter modification. We did not find a reliable distance metric for distributions of f_{age}, which suggests the feature needs experimentation, especially due to the observed impact from modifying α_i and α_s. It is also desirable to capture sensitivity in f_{dyad} for these two parameters. We expect that during fully asymmetric growth, combined with twisting at birth, that the first two cells should migrate into a stacked dyad structure (as seen in Figure 3.2).

We observed the most frequent amount of sensitive features with respect to the three spring force constants k_s, k_i and k_0, as well as the birth twist parameter β. It is clear that the temporal evolution of the scalar features and endpoint distributions for f_{angle}, f_{patch} and f_{age} can be used to formulate a compelling distance comparison between simulations for these four parameters. Note that we can also apply the same metrics used when comparing distributed features for scalar features, if we compare the temporal evolution of the scalar quantities. For example, the absolute integral difference or KL-divergence can be calculated between curves corresponding to different β values in Figure 4.8. A sample result is shown in Supplementary Figure B.4.

We recognize that it's inconclusive whether or not the features used in this sensitivity analysis are optimal or sub-optimal, as there was no succinct sensitivity criterion. Most importantly, we identified which features are promising tools for comparing simulated data using BSim for monoculture growth. Additionally, features which were identified as having no sensitivity with respect to a parameter should not be discounted. All features should be improved further by reducing error and refining the growth modelling.

Other approaches open for investigation include studies by (Yeremi et al., 2014; Koch et al., 2017), which outline a systematic approach to determining the sensitivity and efficacy

of statistical comparison between simulations and observations. In addition to this, a study by (Fearnhead and Prangle, 2012b) suggests an approach to determining optimal summary statistics (features) from data by using fitted linear models as the summary statistics which minimize a quadratic loss function between known and sampled parameters. Although this is a promising method, it may be difficult to translate to agent based simulations.

Chapter 5

Approximate Bayesian Computing

5.1 Overview

The ultimate goal of the feature exploration and sensitivity analysis performed in this study was to collect a number of quantities that can be used to compare two data sets D and D_0, where D is obtained from a mathematical model and D_0 is a real world data set. In our case, the mathematical model we are concerned with is the BSim cell simulation of monoculuture, bacterial growth for a given set of parameters $\theta \in \mathbb{R}^n$, where n is the dimension of our parameter space Θ and D_0 is the data of a target biological system. If there is confidence that the underlying modelling accurately describes the dynamics of the target biological system, then it's assumed there exists some $\theta^* \in \mathbb{R}^n$, such that D is sufficiently similar to D_0 under the same initial conditions. Ideally we would be able to directly compare a data set D given a parameter configuration θ to our experimental data D_0, however, both our model and biological systems are stochastic in nature, making direct comparison impossible. Many parameter inference problems are approached by computing and maximizing the likelihood function $f(D_0|\theta)$, however the likelihood is computationally intractable for stochastic simulations using BSim. We therefore choose to approach finding a θ^* by utilizing the likelihood free methods known as Approximate Bayesian Computing (ABC).

We remove the calculation of a likelihood function in ABC and therefore, the main goal is to approximate a posterior distribution $f(\theta|D_0) \propto f(D_0|\theta)\pi(\theta)$, where $\pi(\theta)$ is a prior distribution for the parameters in our BSim simulations. The fundamental principles of ABC are as follows. A parameter configuration θ' is sampled from a known distribution, and a corresponding simulated data set D' is generated from these parameters. A collection

of p features are calculated for the simulated data and the target data, such that we obtain two feature vectors $\vec{F}_{D'} \in \mathbb{R}^p$ and $\vec{F}_{D_0} \in \mathbb{R}^p$. The parameters θ' are accepted if and only if D' is sufficiently similar to D_0. Similarity between data is determined by a distance metric $d : \mathbb{R}^p \to \mathbb{R}$, where θ' is accepted when $d(\vec{F}_{D'}, \vec{F}_{D_0}) < \varepsilon$ for some tolerance ε. If the feature calculations $\vec{F}$ are sensitive to parameter adjustment, and ε is small, then we can conclude that $f(\theta'|D_0) \approx f(\theta'|d(\vec{F}_{D'}, \vec{F}_{D_0})) < \varepsilon)$. There have been many variations of ABC methods that include rejection sampling (Pritchard et al., 1999; Sunnåker et al., 2013), markov chain monte carlo (MCMC) sampling (Beaumont et al., 2002; Sisson et al., 2007) and sequential monte carlo sampling (SMC) (Del Moral and Jasra, 2007; Toni and Stumpf, 2010; Del Moral et al., 2012). Reviews on the progressions and performance of different ABC methods have also been written by (Marin et al., 2012; Filippi et al., 2013; Lintusaari et al., 2017).

In this study we focus on the ABC-SMC method for approximating the posterior distribution $f(\theta|D_0)$ of BSim parameters. By taking the main features of ABC methods and adding more careful sampling and refinement of the target posterior distribution, we have the following framework for the ABC-SMC method. First, we define a population size of N 'particles', that will comprise samples $\{\theta^{(i)}\}$ for $i = 1, .., N$ from the posterior distribution. Then, for a fixed number of iterations T, we set a schedule for decreasing values of ε such that $\varepsilon_1 \geq \varepsilon_2 \geq ... \geq \varepsilon_T \geq 0$. In theory, if we continue to shrink the accepted distance tolerance between simulated and target data such that $\epsilon_T \approx 0$, the final population $\{\theta_T^{(i)}\}$ should be a highly accurate approximation of $f(\theta|D_0)$. At the end of each iteration t, a weight $w_t^{(i)}$ is calculated for each sample $\theta_t^{(i)}$ to obtain a weighted distribution, represented by the discrete population, that can be sampled from in the next iteration $t + 1$ with probabilities $w_{t-1}^{(i)}$. In the first iteration of ABC-SMC, the parameters are sampled from the prior distribution $\pi(\theta)$ until they are accepted within a tolerance of ε_1 and weights are chosen to be $1/N$. After the first iteration, a perturbation function is applied to the weighted samples $\theta_{t-1}^{(i)}$, to obtain the sample θ, used by the simulation, which is either rejected or accepted according to the distance and epsilon at that iteration. In the ABC-SMC formulation by (Beaumont et al., 2009), a multivariate normal distribution is utilized for the perturbation function with mean $\theta_{t-1}^{(i)}$ and covariance estimated from the previous population θ_{t-1}. Generally the covariance is scaled by a factor of 2 (Beaumont et al., 2009), however, we leave it as an arbitrary constant η. This description of the ABC-SMC method is presented in Algorithm 1.

Algorithm 1 ABC-SMC Beaumont et al. (2009)

Calculate feature vector $\vec{F}_{D_0}$ for target data D_0
Define an epsilon schedule such that $\varepsilon_1 \geq \varepsilon_2 \geq ... \geq \varepsilon_T \geq 0$
Choose scaling factor η for covariance calculations
for $i = 1 \rightarrow N$ **do**
    repeat
        Sample θ' from $\pi(\theta)$
        Generate data D' from parameters θ'
        Calculate feature vector $\vec{F}_{D'}$
    until $d(\vec{F}_{D'}, \vec{F}_{D_0}) < \varepsilon_1$
    $\theta_1^{(i)} = \theta'$
    $w_1^{(i)} = \frac{1}{N}$
end for
$\Sigma_1 = \eta \mathrm{Cov}(\theta_1)$

for $t = 2 \rightarrow T$ **do**
    for $i = 1 \rightarrow N$ **do**
        repeat
            Draw θ'' from previous population θ_{t-1} with probabilities w_{t-1}
            Sample θ' from $\mathcal{N}(\theta'', \Sigma_{t-1})$
            Generate data D' from parameters θ'
            Calculate feature vector $\vec{F}_{D'}$
        until $d(\vec{F}_{D'}, \vec{F}_{D_0}) < \varepsilon_t$
        $\theta_t^{(i)} = \theta'$
        $w_t^{(i)} = \dfrac{\pi(\theta')}{\sum_{j=1}^{N} w_{t-1}^{j} \mathcal{N}(\theta' | \theta_{t-1}^{(j)}, \Sigma_{t-1})}$
    end for
    $\Sigma_t = \eta \mathrm{Cov}(\theta_t)$
end for

5.2 Experimental Setup

We seek to test the ABC-SMC method on inferring the posterior distribution $f(\theta|D_0)$ for the BSim parameters in 2.1. For a simple test case we only consider a three dimensional parameter space that includes the three spring force constants k_i, k_s and k_o and a synthetic data set D_0. A target data set D_0 was generated from our canonical BSim growth simulation, with a total simulation time set to 4 hours using a $dt = 0.02$ hours and initial cell population of 1 cell. The spring force constants were set to $k_i = 50$, $k_o = 500$ and $k_s = 0.01$, with all other BSim parameters set to their default quantities. Utilizing a synthetic data set for the test experiment is crucial in determining the efficacy of data comparison using the results of the feature sensitivity analysis in Section 4.

The distance metric used to reject samples during ABC-SMC is the L2 norm of the feature vector $\vec{F} \in \mathbb{R}^7$. We define the feature vector component by component, depending on the classification of the seven features defined in Section 3. This method of feature vector calculation is used for the ABC-SMC simulations and the synthetic data simulation. For each of the four scalar features f_{density}, f_{dyad}, f_{ap} and f_{anis} we only calculate their quantity at the final time step of the simulation. This was chosen to reduce computations each time a BSim simulation is invoked during the ABC-SMC inference. After obtaining the endpoint value for each feature, we then set the first four elements of $\vec{F}$ to the corresponding scalar feature. For the three distributed features f_{angle}, f_{patch} and f_{age} we chose to reduce each distribution to a scalar quantity that can be assigned to the final three elements of the feature vector $\vec{F}$. This was accomplished by numerically integrating the distributions using the Simpson method and setting the last three elements of the feature vector to the corresponding integral quantity. We can then easily perform this sequence of calculations for a simulated data set D' and target data set D_0 in order to calculate the distance $d(\vec{F}_{D'}, \vec{F}_{D_0})$ between the two data sets where,

$$d(\vec{F}_{D'}, \vec{F}_{D_0}) = ||\vec{F}_{D'} - \vec{F}_{D_0}||_2. \tag{5.1}$$

To perform the ABC-SMC calculations, we utilized the open source Python library *pyabc* (Klinger et al., 2018). *pyabc* offers many tools to perform and customize ABC-SMC calculations, however, by default it will run Algorithm 1 with a scaling factor of $\eta = 1$. Beyond the default configuration in *pyabc*, the user must supply the program with a user defined model and distance function, as well as the parameter prior distributions $\pi(\theta)$, particle population size N, maximum number of iterations T and an epsilon schedule. In our implementation of ABC-SMC using *pyabc*, we utilized BSim as our model and the distance function in Equation 5.1, with additional configurations outlined in Table

5.1. *pyabc* also offers many parallel sampling methods which allow concurrent simulations, thus increasing computational speed. In our test ABC-SMC experiment we utilized the HPC resources on the ComputeCanada cluster Graham to run 10 parallel simulations. The Python code and documentation for this ABC experiment can be found on GitHub https://github.com/ingallslab/bsim-hpc-package.

pyabc Option	Choice
Prior Distributions $\pi(\theta)$	$k_i \rightarrow$Uniform(0,100) $k_o \rightarrow$Uniform(0,1000) $k_s \rightarrow$Uniform(0,1)
Population Size N	100
Maximum Iterations T	20
Epsilon Schedule	ε_t is set to median $d(\vec{F}_{D'}, \vec{F}_{D_0})$ at iteration $t-1$. $\varepsilon_{\min} = 0.1$.

Table 5.1: ABC-SMC configuration for implementation using the Python library *pyabc*

5.3 Results

Despite constricting the ABC-SMC experiment to a small parameter space, the ABC-SMC experiment outlined in Section 5.2 yielded sub-optimal results. Figures 5.1, 5.2 and 5.3 each show a similar story that the estimation of each parameter's one dimensional posterior is struggling to converge around the exact value used to simulate D_0. Although the results suggest that the feature calculations and distance metric outlined in Section 5.2 are insufficient for accurate posterior estimation, the data is inconclusive to make this conclusion. It's suspected that poor results occurred mainly due to computational challenges related to the ABC-SMC inference of BSim parameters. The maximum number of iterations (populations) that the experiment was intended to run for was $T = 20$, however, for a population size of $N = 100$ and corresponding BSim simulation configuration, the full number of iterations was never attained. This was due to computational resources becoming unavailable after the requested time limit of 48 hours was surpassed, thus producing a result that never fully completed the full ABC-SMC algorithm. It's likely that a population size of 100 is also too small to accurately infer the posterior distribution.

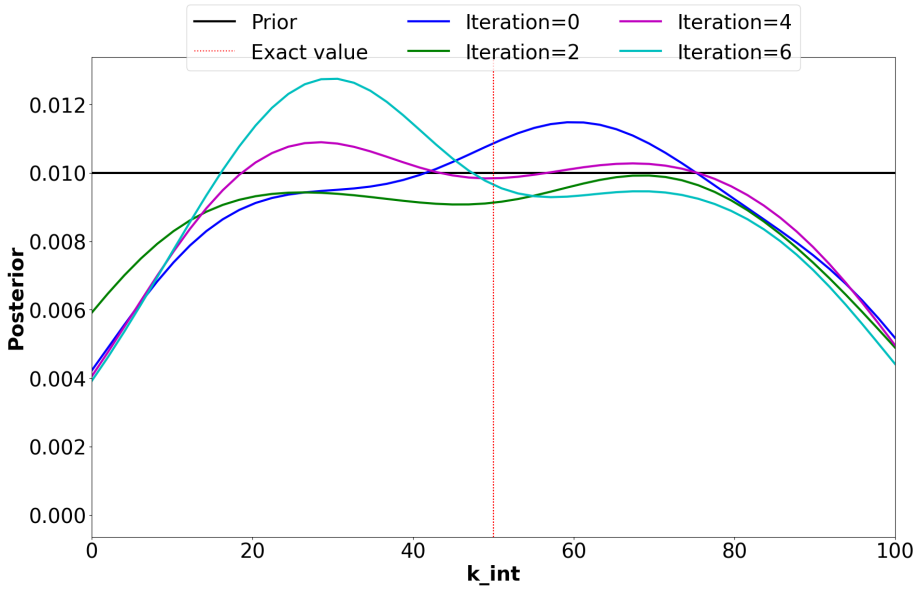

Figure 5.1: One dimensional KDE plot of internal spring force constant k_i posterior distribution inferred by ABC-SMC.

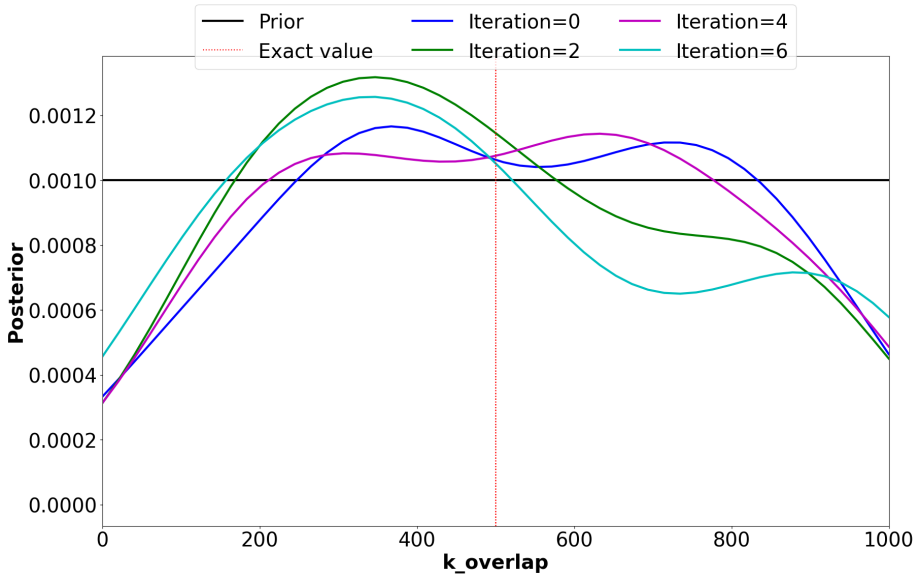

Figure 5.2: One dimensional KDE plot of overlap spring force constant k_o posterior distribution inferred by ABC-SMC.

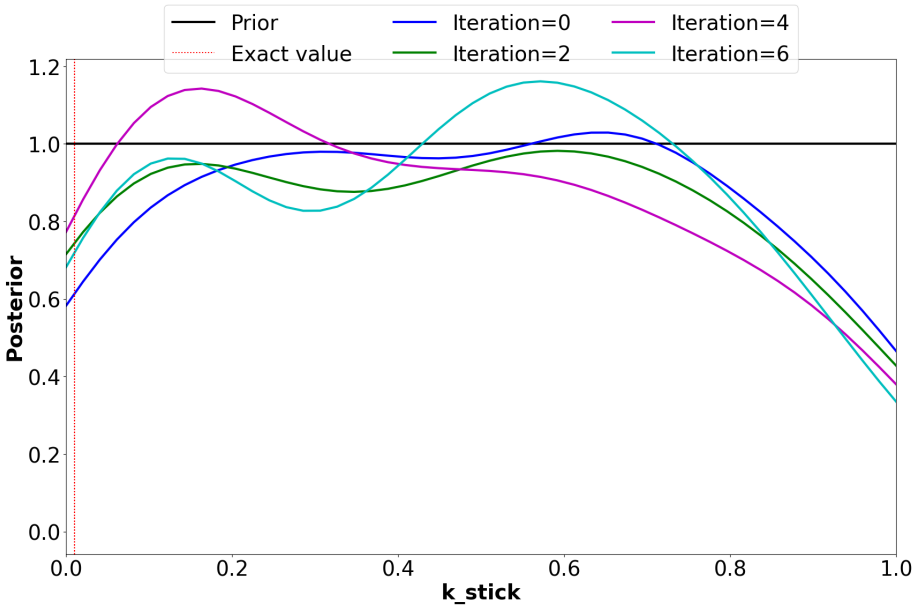

Figure 5.3: One dimensional KDE plot of sticking spring force constant k_s posterior distribution inferred by ABC-SMC.

5.4 Discussion

Setting up ABC-SMC runs of the scale expected to produce adequate results poses an interesting problem for many reasons. Increasing the number of particles per population and size of the parameter space subsequently increase the time required to complete an ABC-SMC calculation such that $d(\vec{F}_{D'}, \vec{F}_{D_0}) < \epsilon_T \approx 0$. Additionally, creating an epsilon schedule such that ε approaches 0 too quickly and too close to 0, will likely cause the algorithm run indefinitely. This is due to the acceptance rate of particles being extremely small when ε is small. For example, in our ABC-SMC experiment we observed that by the sixth iteration, the acceptance rate had dropped to around 1%, which results in the sampling and computation of $\sim 100N$ simulations to satisfy ε_6 (shown in Table 5.2). With each BSim simulation running for a minimum of 20 seconds given the initial conditions and simulation set up, that amounts to 5 hours of computation for the sixth iteration given that we are able to run 10 concurrent simulations. It is important to note that although HPC offers resources suitable for running long computations, an $\varepsilon_T \approx 1$ and $N = 100$ are not suitable for producing accurate results and the compute time will increase proportionally as your decrease ε_t and increase N. Therefore, it is evident that there is a careful trade-off

that must occur when balancing total simulation time, epsilon scheduling, population size and the available computing resources.

Iteration t	ε_t	Acceptance Rate
0	-	100.0 %
1	90.0	38.4 %
2	46.0	25.5 %
3	23.2	15.1 %
4	11.0	6.4 %
5	6.0	2.6 %
6	4.2	1.6 %

Table 5.2: Resulting epsilon schedule from ABC-SMC experiment, with corresponding particle acceptance rates per ε_t in percentage.

Beyond configuring the ABC-SMC algorithm and BSim simulations, there a number of potential modifications that can be made to improve performance. First, utilizing more results from the sensitivity analysis in this study to craft a more dense feature vector $\vec{F}$ should be explored. By dense, we imply that the feature vector should contain as much information as possible from the available temporal evolution of each feature defined in Section 3. An immediately obvious extension from our ABC-SMC experiment would be to include the temporal evolution of each of the scalar features (excluding dyad structure). By including the full set of points or distributions for each feature, extra care must then be taken to construct a distance metric that properly handles each feature type directly. For example, the Wasserstein distance metric could be applied to each of the temporal and distributed features individually and then averaged over the resulting distances.

Additional techniques have been proposed in ABC literature to improve performance. These studies include adaptive distance metrics (Fearnhead and Prangle, 2012a; Prangle, 2017) and post sampling correction methods that fit linear or non-linear models to intermediate posterior distributions (Lintusaari et al., 2017). Both of these methods are included in *pyabc*. The reader should also refer to (Lintusaari et al., 2017; Grazzini et al., 2017; Sisson et al., 2018) for more information on recent advances and methods in boosting ABC performance.

Chapter 6

Conclusion

In this study we have proposed a systematic approach to determining features for data comparison between stochastic agent based simulations of monocultural, microbial communities and experimental data. We performed a one at a time sensitivity analysis for features inspired by several studies (You et al., 2018; Dell'Arciprete et al., 2018; Doumic et al., 2020) with respect to the relevant BSim parameters that dictate the dynamics of our growth simulation. It was found that most features had meaningful responses to 2 or more parameters, therefore indicating that the features could act as suitable tools for measuring quantitative distances between simulation and data. Parameters that showed little to no feature sensitivity should be investigated further as their parameter ranges may have been outside of the range of sensitivity of the studied features. Alternatively, the model parameterizations may not be implemented such that they cause significant impacts on the dynamics of simulated cell colony growth. Model refinement should be explored further to ensure errors are reduced in feature calculations, as well as reducing errors in the target model behaviour.

Utilizing the features investigated in the one at a time sensitivity analysis, we constructed a simple L2 distance metric which captured the distance between synthetic BSim data set and other simulated data sets. The distance metric compared features at a single time point of the data and should be extended to the full temporal structure of the feature (where possible). Finally, we performed a test to infer the posterior distributions for the parameters used in our BSim growth simulations, where a synthetic data set was crafted as the target. Our inference calculations were performed using the ABC-SMC algorithm, employing the l2 distance metric. The test was unable to infer accurate posterior distributions for the known parameters used in the simulation. This was likely due to insufficient scaling of the population size and computational time used in the ABC-SMC run. Given

this was the inaugural run for parameter inference in our BSim growth simulations, results are likely to be refined after further tests are performed.

References

G An, · B G Fitzpatrick, · S Christley, · P Federico, · A Kanarek, · R Miller Neilan, · M Oremland, · R Salinas, · R Laubenbacher, and · S Lenhart. Optimization and Control of Agent-Based Models in Biology: A Perspective. *Bull Math Biol*, 79:63–87, 2017. doi: 10.1007/s11538-016-0225-6.

Saeed Asaeedi, Farzad Didehvar, and Ali Mohades. Alpha-Concave Hull, a Generalization of Convex Hull.

Mark A. Beaumont, Wenyang Zhang, and David J. Balding. Approximate Bayesian Computation in Population Genetics. *Genetics*, 162(4), 2002.

Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P Robert. Adaptive approximate Bayesian computation. *Source: Biometrika*, 96(4):983–990, 2009.

Steven A Benner and A Michael Sismour. Synthetic biology. *Nature Reviews Genetics*, 6 (7):533–543, 2005.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

D Ewen Cameron, Caleb J Bashor, and James J Collins. A brief history of synthetic biology. *Nature Reviews Microbiology*, 12(5):381–390, 2014.

Mark R Charbonneau, Vincent M Isabella, Ning Li, and Caroline B Kurtz. Developing a new class of engineered live bacterial therapeutics to treat human diseases. *Nature communications*, 11(1):1–11, 2020.

Ye Chen, Jae Kyoung Kim, Andrew J Hirning, Krešimir Josić, and Matthew R Bennett. Emergent genetic oscillations in a synthetic microbial consortium. *Science*, 349(6251): 986–989, 2015.

Alex Clark. Pillow (pil fork) documentation, 2015.

Pierre Del Moral and Ajay Jasra. Sequential Monte Carlo for Bayesian Computation. *Bayesian Statistics*, 8:1–34, 2007.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.

D. Dell'Arciprete, M. L. Blow, A. T. Brown, F. D.C. Farrell, J. S. Lintuvuori, A. F. McVey, D. Marenduzzo, and W. C.K. Poon. A growing bacterial colony in two dimensions as an active nematic. *Nature Communications*, 9(1):1–9, oct 2018.

Marie Doumic, Sophie Hecht, and Diane Peurichard. A purely mechanical model with asymmetric features for early morphogenesis of rod-shaped bacteria micro-colony. *Mathematical Biosciences and Engineering*, 17(6):6873–6908, 2020.

Giorgio Fagiolo, Mattia Guerini, Francesco Lamperti, Alessio Moneta, and Andrea Roventini. Validation of agent-based models in economics and finance. In *Computer simulation validation*, pages 763–787. Springer, 2019.

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation. *J. R. Statist. Soc. B*, 74:419–474, 2012a.

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, 2012b.

Lucas Fievet and Didier Sornette. Calibrating emergent phenomena in stock markets with agent based models. *PloS one*, 13(3):e0193290, 2018.

Sarah Filippi, Chris P. Barnes, Julien Cornebise, and Michael P.H. Stumpf. On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo. *Statistical Applications in Genetics and Molecular Biology*, 12(1):87–107, 2013.

Timothy S Gardner, Charles R Cantor, and James J Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, 2000.

Narendra S Goel and Nira Richter-Dyn. *Stochastic models in biology*. Elsevier, 2016.

José Eduardo González-Pastor, Diego Francisco Romero, Pascale Beauregard, Paul D Straight, Reed M Stubbendieck, and Carol Vargas-Bautista. Bacterial Communities: Interactions to Scale. 2016.

Thomas E. Gorochowski. Agent-based modelling in synthetic biology. *Essays in Biochemistry*, 60(4):325–336, 2016.

Jakob Grazzini, Matteo G Richiardi, and Mike Tsionas. Bayesian estimation of agent-based models. *Journal of Economic Dynamics and Control*, 77:26–47, 2017.

D M Hamby. A Review of Techniques for Parameter Sensitivity. *Environmental Monitoring and Assessment*, 32(c):135 –154, 1994.

Florian Hartig, Justin M. Calabrese, Björn Reineking, Thorsten Wiegand, and Andreas Huth. Statistical inference for stochastic simulation models - theory and application, aug 2011.

David G. Kirkpatrick and Raimund Seidel. On the Shape of a Set of Points in the Plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

Emmanuel Klinger, Dennis Rickert, and Jan Hasenauer. pyabc: distributed, likelihood-free inference. *Bioinformatics*, 34(20):3591–3593, 2018.

Eric W. Koch, Caleb G. Ward, Stella Offner, Jason L. Loeppky, and Erik W. Rosolowsky. Identifying tools for comparing simulations and observations of spectral-line data cubes. *Monthly Notices of the Royal Astronomical Society*, 471(2):1506–1530, oct 2017.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

David T Kysela, Pamela J B Brown, Kerwyn Casey Huang, and Yves V Brun. Biological Consequences and Advantages of Asymmetric Bacterial Growth Morphological asymmetry: cell division that produces two daughter cells with distinct shapes. 2013.

Juliane Liepe, Paul Kirk, Sarah Filippi, Tina Toni, Chris P Barnes, and P H Michael. A Framework for Parameter Estimation and Model Selection from Experimental Data in Systems Biology Using Approximate Bayesian Computation. *Nature Protocols*, 9(2): 439–456, 2014.

Jarno Lintusaari, Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, 66(1):e66–e82, 2017.

Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

Antoni Matyjaszkiewicz, Gianfranco Fiore, Fabio Annunziata, Claire S. Grierson, Nigel J. Savery, Lucia Marucci, and Mario di Bernardo. Bsim 2.0: An advanced agent-based cell simulator. *ACS Synthetic Biology*, 6(10):1969–1972, 2017. PMID: 28585809.

William Paranchych and Laura S. Frost. The Physiology and Biochemistry of Pili. *Advances in Microbial Physiology*, 29(C):53–114, jan 1988.

Bruno Pietzsch, Sebastian Fiedler, Kai G Mertens, Markus Richter, Cédric Scherer, Kirana Widyastuti, Marie-Christin Wimmler, Liubov Zakharova, and Uta Berger. Metamodels for evaluating, calibrating and applying agent-based models: a review. *Journal of Artificial Societies and Social Simulation*, 23(2), 2020.

Vitor B. Pinheiro and Thomas E. Gorochowski. Agent-based modelling in synthetic biology. *Essays in Biochemistry*, 60(4):325–336, 11 2016.

Dennis Prangle. Adapting the ABC Distance Function. *https://doi.org/10.1214/16-BA1002*, 12(1):289–309, mar 2017.

Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun, and Marcus W Feldman. Population Growth of Human Y Chromosomes: A Study of Y Chromosome Microsatellites. *Mol. Biol. Evol*, 16(12):1791–1798, 1999.

Marc-Sven Roell and Matias D Zurbriggen. The impact of synthetic biology for future agriculture and nutrition. *Current opinion in biotechnology*, 61:102–109, 2020.

Gregory D Sepich-Poore, Laurence Zitvogel, Ravid Straussman, Jeff Hasty, Jennifer A Wargo, and Rob Knight. The microbiome and human cancer. *Science*, 371(6536), 2021.

S. A. Sisson, Y. Fan, and Mark M. Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, feb 2007.

Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC Press, 2018.

Gregory Stephanopoulos. Synthetic biology and metabolic engineering. *ACS synthetic biology*, 1(11):514–525, 2012.

Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate Bayesian Computation. *PLOS Computational Biology*, 9(1):e1002803, jan 2013.

Tzu-Chieh Tang, Bolin An, Yuanyuan Huang, Sangita Vasikaran, Yanyi Wang, Xiaoyu Jiang, Timothy K Lu, and Chao Zhong. Materials design by synthetic biology. *Nature Reviews Materials*, 6(4):332–350, 2021.

Tina Toni and Michael P. H. Stumpf. Simulation-based model selection for dynamical systems in systems and population biology. *Bioinformatics*, 26(1):104–110, jan 2010.

Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

Pauli et. al. Virtanen. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Brian J Yeh and Wendell A Lim. Synthetic biology: lessons from the history of synthetic organic chemistry. *Nature chemical biology*, 3(9):521–525, 2007.

Miayan Yeremi, Mallory Flynn, Stella Offner, Jason Loeppky, and Erik Rosolowsky. Comparing simulated emission from molecular clouds using experimental design. *Astrophysical Journal*, 783(2), 2014.

Zhihong You, Daniel J G Pearce, Anupam Sengupta, and Luca Giomi. Geometry and Mechanics of Microdomains in Growing Bacterial Colonies. *Physical Review X*, 8, 2018.

APPENDICES

Appendix A

Supplementary Figures

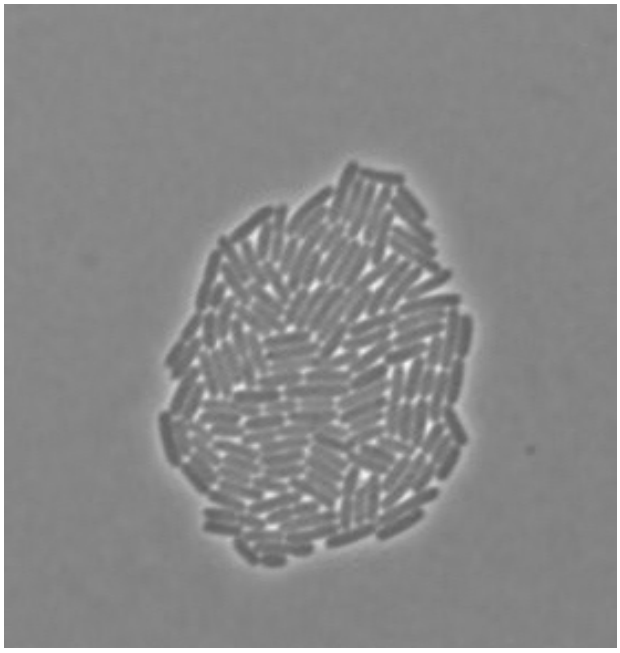

Figure A.1: Sample image of a single *E. coli* cell colony obtained from phase microscopy imaging.

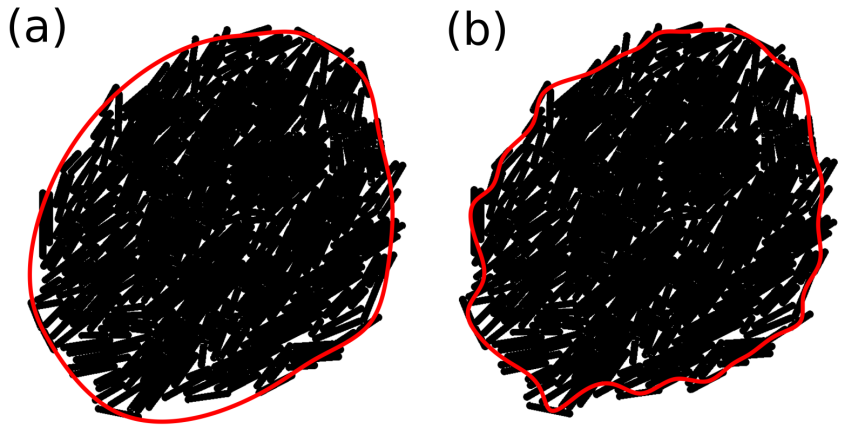

Figure A.2: (a) Resulting cubic spline fitted to alphashape points obtained for an $\alpha = 0$, which corresponds to calculating the convex hull. (b) Resulting cubic spline fitten to alphashape points obtain for an $\alpha = 0.01$.

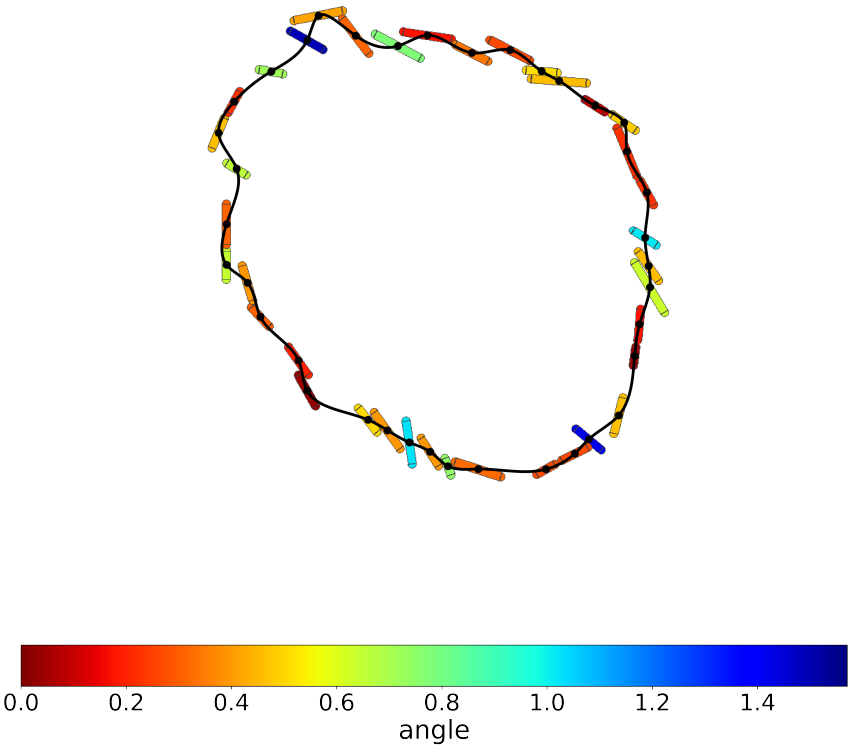

Figure A.3: Sample calculation output for cell orientation with respect to cell colony boundary. Boundary is calculated using cubic spline fitted to alphashape with $\alpha = 0.01$.

Appendix B

Sensitivity Analysis Results

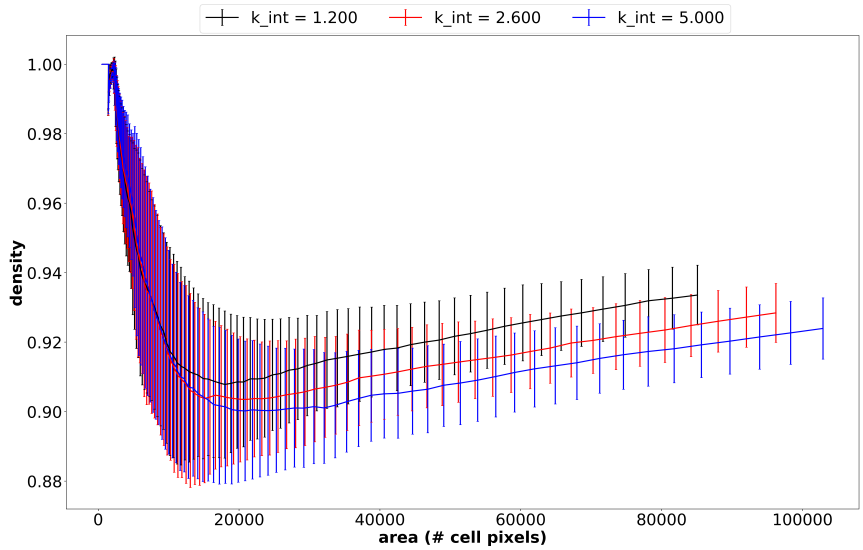

Figure B.1: Mean density (f_{density}) as a function of cell colony area for three different internal force constant k_i simulation ensembles. Error bars represent standard deviation of the density over the simulation ensemble.

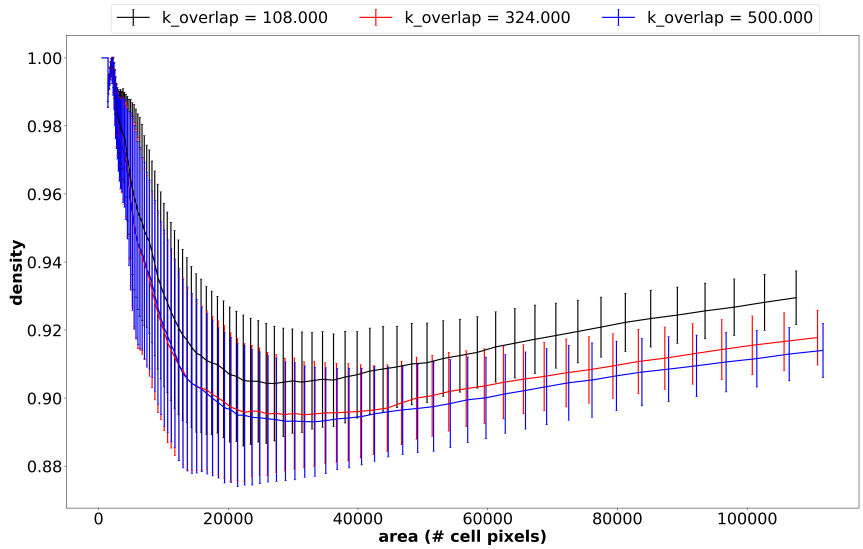

Figure B.2: Mean density (f_{density}) as a function of cell colony area for three different overlap force constant k_o simulation ensembles. Error bars represent standard deviation of the density over the simulation ensemble.

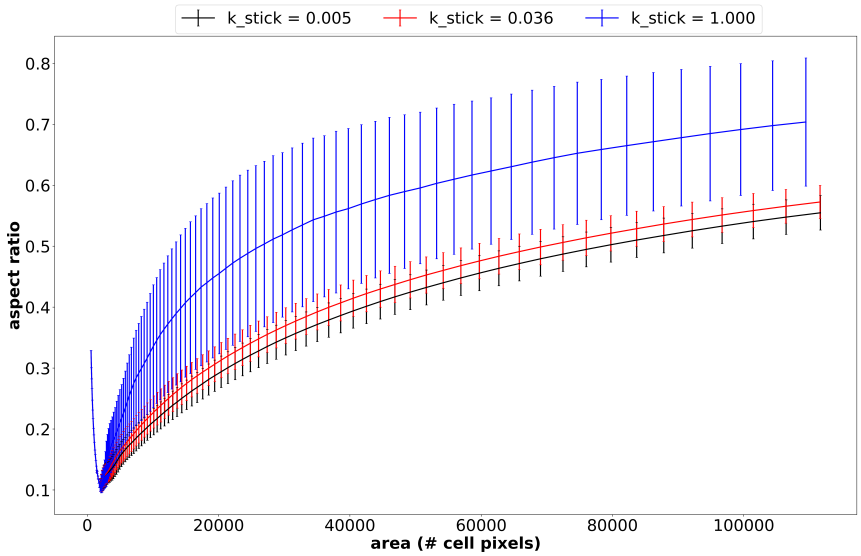

Figure B.3: Mean density (f_{ap}) as a function of cell colony area for three different sticking force constant k_s simulation ensembles. Error bars represent standard deviation of the density over the simulation ensemble.

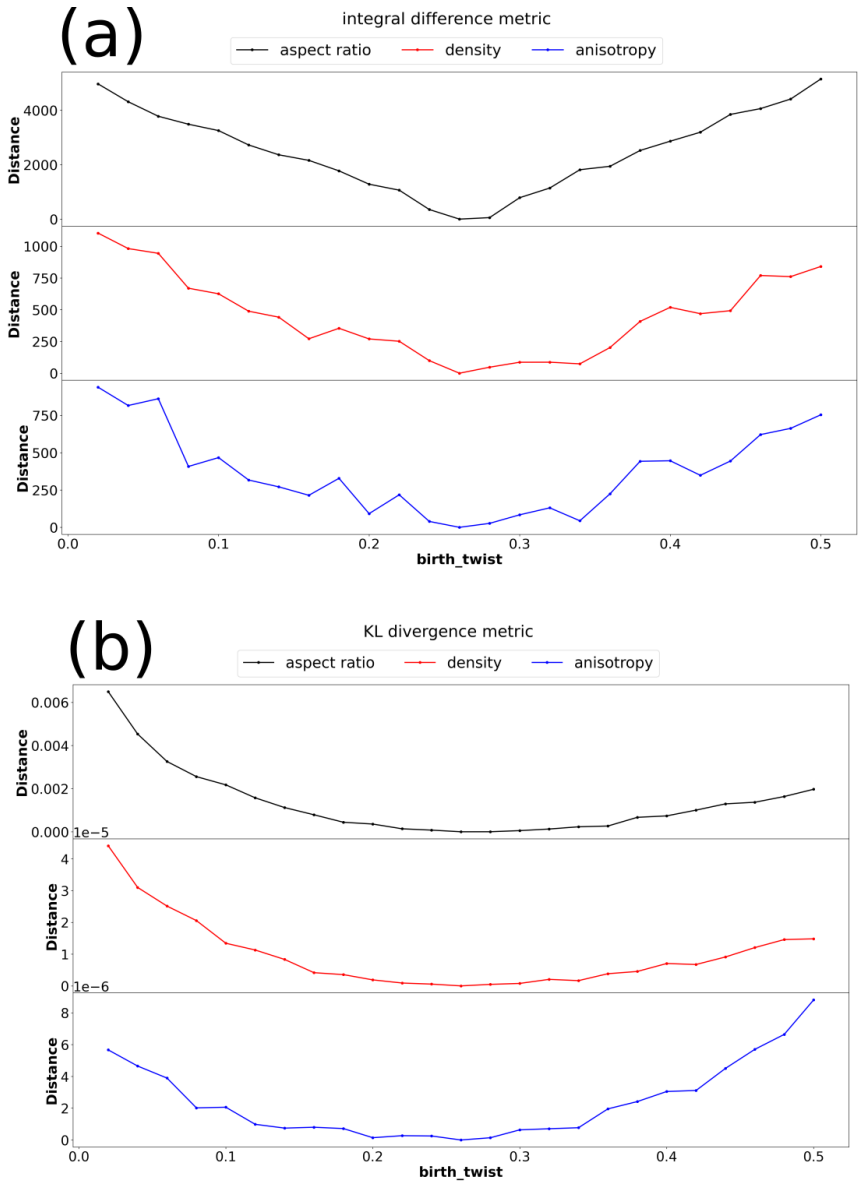

Figure B.4: The distance between scalar feature values as a function of colony area are calculated for BSim simulation ensembles for birth twist parameter β. (a) The absolute integral difference is obtained between temporal evolution of aspect ratio, density and anisotropy. (b) The KL divergence is obtained between temporal evolution of aspect ratio, density and anisotropy.