# Topological Data Analysis of Embryonic Pluripotent Stem Cells

Analyzing Persistent Homology in Lab-Obtained Images of Embryonic Stem Cell Clusters and Developing an Agent-Based Model for Comparison

Jaxon Green

September 9, 2024

APPROVAL PAGE

| | |
|---|---|
| TITLE: | Topological Data Analysis of Embryonic Pluripotent Stem Cells |
| AUTHOR: | Jaxon Green |
| DATE SUBMITTED: | June 2022 |

| | |
|---|---|
| Senior Project Advisor | Signature |
| Mathematics Department Chair | Signature |

# Contents

# 1 Introduction

Understanding the potential for the use of stem cells as regenerative medical treatment of disease has long been a challenging endeavor for clinical researchers. In the last decade, the scientific community has overcome tremendous hurdles and made incredible progress towards unraveling the possibilities in stem-cell-based medicine. However, there are still many barriers that scientists face today, preventing them from obtaining broadly applicable techniques that can be generalized to an entire population in need of aid. As a result, the emergence of new ways of identifying and analyzing cells and their regenerative properties must be developed.

A persistent focal point in stem cell research is the self-organization of cells via cell-cell communication in-vivo. As stem cells transition from their pluripotent stage into differentiated, structurally ordered tissues, we can analyze and model patterns in cell kinematics to develop more refined axioms that govern cell behavior. This environment not only proves as a hub for analyzing stem cell conduct but also for understanding coordinated actions of generalized cells in life-like conditions.

Methods have previously been developed for quantifying self-organization, but are limited to manual inspection or global measures in many application. In specific, machine-learning algorithms, such as convolutional networks, those have consistently failed to illuminate mechanisms governing local behavior of cells and continue to remain a computationally expensive operation. This project explores topological data analysis (TDA) as a means to assign a metric to patterns observed in images of stem cell clusters.

# 2 Biology of Human Embryonic Pluripotent Stem Cells

## 2.1 In-Vivo Development

Stem cells are the first stage in human embryonic development. From the moment that a sperm fertilizes an egg, two haploid gametes combine into one new diploid zygote. This zygote is the very first pluripotent stem cell. Many cycles of mitosis will split and replicate this zygote in order to create a blastocyst. The structure of a blastocyst largely consists of a mostly hallow ball of rapidly dividing stem cells. The outer layer of this ball structure will develop into the placenta and umbilical chord in-vivo. Within the blastocyst

lies the embryoblast, or the Inner Cell Mass (ICM). Within the ICM, the epiblast is what will develop and evolve into the human embryo. This project focuses on the emerging patterns observed in the epiblast, during a process called gastrulation, as stem cells self-organize and influence one another to ultimately differentiate. In human development, there are three germ layers within the ICM which form based on stem cell differentiation. The three layers consist of the endoderm, mesoderm, and ectoderm. The endoderm, the inner-most germ layer, will eventually develop to become the digestive and respiratory tract in the human body. The mesoderm, the intermediate germ layer, will eventually become the muscular and skeletal systems. The ectoderm, the outer-most germlayer, will develop into the nervous system and epidermal layers. Cell differentiation occurs almost entirely based on each stem cells location in the ICM. Understanding what governs cell movement in-vivo is critical to understanding and predicting cell fate. [1, 2, 3]

Figure 1 depicts the stages of embryonic development leading up to the development of the embryoblast.

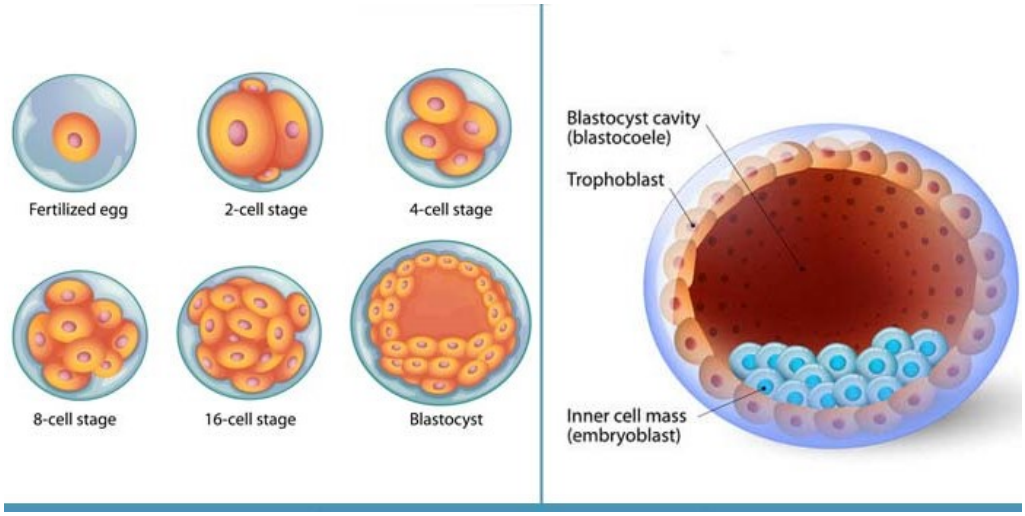



Figure 1: Stages of Embryonic Development

## 2.2 BMP4-NOG Reaction Diffusion

In-vivo, stem cells naturally emit specialized proteins called morphogens into their local environment. These proteins create asymmetric patterning in large clusters of cells as they use chemical signalling mechanisms to direct

cell motion. It is thought that there exist hundreds of these tiny particles in the cellular medium that preform different tasks, but among the most important are the BMP4 and Noggin (NOG) morphogen pair.

This biochemical process of influencing cell interactions can be best understood through a reaction diffusion model. A reaction diffusion model is defined by two molecules, an activator and an inhibitor. In our case, the activator protein is BMP4 and the inhibitor is NOG. Each of these chemicals have their own respective diffusivities and therefore move at different speeds in the medium. In-vivo, these morphogens freely roam in the extracellular medium, however when a BMP4 molecule encounters a NOG molecule, whether by collision of chemical signaling, the BMP4 molecule becomes inhibited. In other words, BMP4 motion is hindered by the geometry of the NOG molecules as well as BMP4 singaling capability is dampened for a significant amount of time. As a result, we see relatively concentrated amounts of BMP4 in certain locations that cannot diffuse beyond the confines of their imprisonment by NOG proteins.

In Figure 2, we can see that BMP4 is more active at the edges of our circular medium. This is evidence that NOG inhibits BMP4 from diffusing into the internal cells, limiting BMP4 exposure in high capacities to the outermost cells. pSMAD1 is a signalling protein emitted by BMP4, and thus we see the positional location of both BMP4 and pSMAD1 must be relatively confined to the same locations.

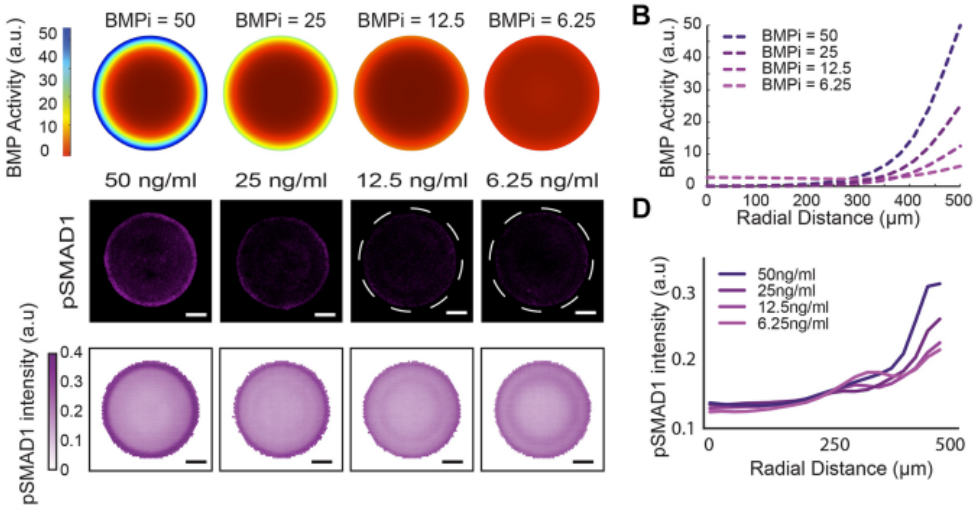



Figure 2: Concentration of BMP4 and corresponding emitted pSMAD1 invitro

The mechanics of this reaction diffusion network leads to two key observations. First, the amount of concentrated exposure that stem cells get to uninhibited BMP4 is a direct indication of which germ layer a stem cell will differentiate into. Due to the nature of clustering of BMP4 molecules, we can observe a relatively symmetric differentiation gradient forming in an otherwise asymmetric cluster of cells. Second, pockets of BMP4 that are isolated due to its interaction with NOG will influence stem cell motion, gravitating them towards their position in the medium, ultimately leading to the creation of pockets and cavities in the original epiblast. These emerging patterns are ideal for analysis using TDA tools. [4, 5]

In Figure 3, we can see different germ layers that emerge as a result of exposure to different concentrations of BMP4. Blue corresponds to endoderm cells, green to mesoderm cells, and red to ectoderm cells.

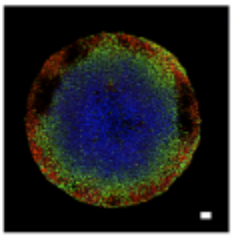


Figure 3: Stem Cell Differentiation Gradient

# 3 Topological Data Analysis

In layman's terms, topological data analysis is the practice of searching for shapes in large data sets. Using abstract topological concepts, mathematicians can optimize their approach to data analysis. TDA allows us to ignore noisy data that would otherwise skew insight gained by a potential statistical analysis, and hone in on the most relevant and persistent features of a data set.

All of the definitions, images, and most examples in Section 3 are taken from summer research done with Dr. Elena Dimitrova.

## 3.1 Simplicial Complexes

Before we can understand how to find "shapes in the data" we must define what a shape is. In order to explore the notion of a simplicial complex, we must understand the notion of convexity.

**Definition 3.1** (Convex set)**.** A subset of a Euclidean Space (or more generally an affine space over $\mathbb{R}$) is convex if, for all pairs of points in the subset, the subset contains the whole line segment that joins them.

**Example 3.2.** The set $\{(x, y, z) \in \mathbb{R}^3 \mid z = 1\}$ is a convex set

**Definition 3.3** (Convex hull)**.** The convex hull of a shape is the smallest convex set that contains it. In other words, it is the intersection of all convex sets containing a given subset of a Euclidean space.

**Example 3.4.** Consider the unit sphere in $\mathbb{R}^3$,

$$S = \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 = 1\}$$

The convex hull of $S$ is the unit ball in $\mathbb{R}^3$

$$B = \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 \leq 1\}$$

**Definition 3.5** (Simplex)**.** A k-simplex is the convex hull of a set $P$ of $k+1$ affinely independent points. A k-simplex is said to have dimension $k$.

This is what we define to be our "shape". We do not always have the luxury of being guaranteed something familiar, like a triangle, that will pop out at us in a data set. With this definition, we encapsulate all the polygons we know and love as well as more.


Figure 4: Examples of $k$-dimensional simplexes

**Definition 3.6** (Face)**.** Consider a simplex, $\sigma$. A face of $\sigma$, is a simplex that is the convex hull of a nonempty subset of $P$. Faces of $\sigma$ come in all dimensions from zero ($\sigma$'s vertices) to k; $\sigma$ is a face of $\sigma$. A proper face of $\sigma$ is a simplex that is the convex hull of a proper subset of $P$; i.e. any face except $\sigma$. In particular, the $(k-1)$-faces of $\sigma$ are called facets of $\sigma$; $\sigma$ always has $k+1$ facets.

**Example 3.7.** In the 3-simplex in the previous image, we can see that there are 4 2-simplexes that compose each facet of the shape.

If we consider to represent a simplex as a set of size $k+1$, each element corresponding to a connected point, we can consider an $n$-face to be a subset of size $n+1$.

**Definition 3.8** (Abstract Simplicial Complex)**.** A collection $\mathcal{A}$ of subsets of a given set $A$ is an Abstract Simplicial Complex if every element $\sigma \in \mathcal{A}$ has all of its subsets $\sigma' \subseteq \sigma$ also in $\mathcal{A}$. The elements of $\mathcal{A}$ are vertices of $\mathcal{A}$. Each (sub)set in $\mathcal{A}$ is a simplex whose dimension equals its cardinality.

In other words, given a collection of data, we can define an abstract simplicial complex to be a set of shapes and all corresponding, lower dimensional components that construct our original shape.

**Example 3.9.** Consider the set $A = \{a, b, c, d, e, f\}$ and the simplicial complex $\mathcal{A} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{b, c\}, \{c, d\}, \{ce\}, \{d, e\}, \{c, d, e\}, \{d, f\}, \{e, f\}\}$. This simplicial complex is visualized in Figure 5.

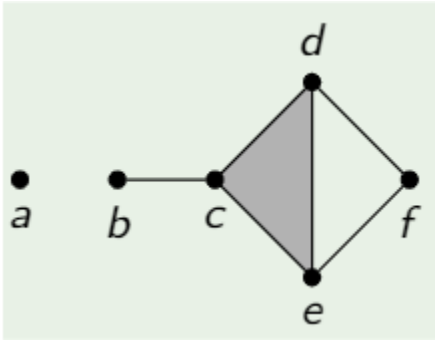



Figure 5: The simplicial complex from Example 3.9

Now, in order to analyze shapes from a data set, we need to define some parameters regarding what kind of shapes we are looking to analyze.

**Definition 3.10** (Metric space)**.** A metric space is a pair $(M, d)$ where $M$ is a set and $d$ is a distance function $d : M \times M \to R$ satisfying the following properties:

- $d(x, y) \geq 0$ for all $(x, y) \in M \times M$

- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ for all $(x, y) \in M \times M$
- $d(x, y) \leq d(x, z) + d(z, y)$ for all $(x, y, z) \in M \times M \times M$

**Example 3.11.** $(\mathbb{R}^n, d(x, y))$ is a metric space where $d(x, y)$ is the standard Euclidean distance

We can use the notion of an abstract metric in order to satisfy our own needs. In our case, we can just consider our metric to be simply the Euclidean distance formula.

**Definition 3.12** (Vietoris-Rips Complex)**.** Let $(P, d)$ be a metric space where $P$ is a point set. Given a real $r \geq 0$, the Vietoris-Rips (or just Rips in short) Complex is the abstract simplicial complex $R^r(P)$ where a simplex $\sigma \in R^r(P)$ if and only if $d(p, q) \leq r$ for every pair of vertices of $\sigma$.

**Example 3.13.** In Figure 6 above, we see what the Rips-Complex looks like for multiple values of $r$. As our parameterized value of $r$ increases, the Rips-Complex contains more higher-dimensional simplexes. When $r$ reaches a large enough value, every possible simplex that can be created from the data given is in the Rips-Complex

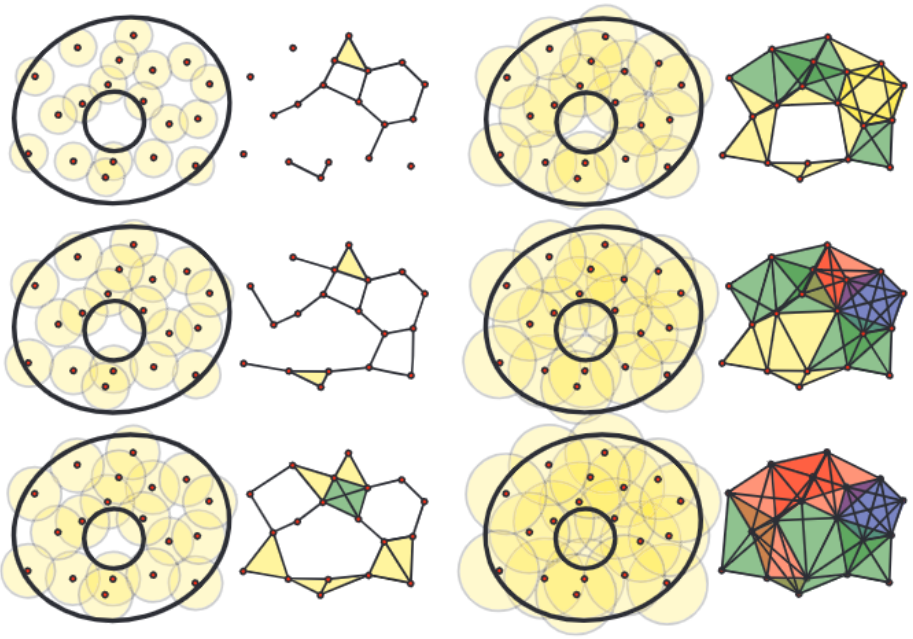


Figure 6: Point Clouds with corresponding Rip-Complex given a radius value

This is our tool for analysis. With the Rips complex, we parameterize the radius form the center of all points simultaneously and create a simplicial complex based on overlapping radii. The composition of this simplicial complex will depend on our data set and our radius value. This simplicial complex will characterize topological features in a data set based on our metric. In particular, we focus our attention to the topological features corresponding to connected-ness of components and to loops. Respectively, these correspond to 0-dimensional homology and 1-dimensional homology. The set of p-dimensional homologies is denoted for a given data set, $A$ is $H_p(A)$.

## 3.2 Persistent Homology

In order to make more intuitive sense of the Rips-Complex, we introduce the concept of a filtration.

**Definition 3.14** (Filtration)**.** A filtration of a simplicial complex, $K$, is a nested sequence of subcomplexes starting at the empty set and ending with

the full simplicial complex, i.e.,

$$\emptyset \subset K_0 \subset K_1 \subset \cdots \subset K_m = K.$$

**Example 3.15.** Let $K = \{\{abc\}, \{ac\}, \{bc\}, \{ab\}, \{a\}, \{b\}, \{c\}, \emptyset\}$. Then, a possible filtration of K would be

$$\emptyset \subset \{\{a\}, \emptyset\} \subset \{\{a\}, \{b\}, \{c\}, \emptyset\} \subset \{\{ab\}, \{a\}, \{b\}, \{c\}, \emptyset\} \subset K$$

If we consider the Rips-Complex with sufficiently large $r$ to be our $K_m$, we can construct a filtration of $K_m$ in such a way that the nested subsets correspond to Rip-Complexes with different values of $r$. We can characterize topological features of our data set by when they first appear in the filtration and when they disappear.

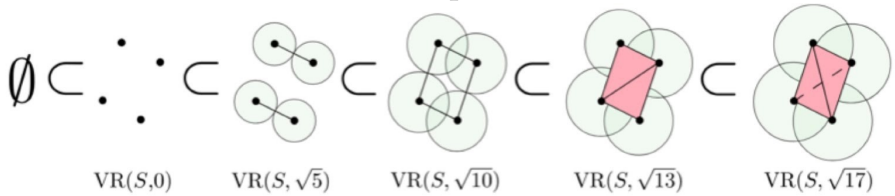
**Definition 3.16** (Birth)**.** For a filtered complex, $K$, and subcomplexes $K_i$,$K_j$ where $i < j$, a topological feature $x \in H_p(K_j)$ is born at $j$ if $x \notin H_p(K_i)$.

**Definition 3.17** (Death)**.** For a filtered complex $K$ and subcomplexes $K_i$,$K_j$ where $i < j$, a topological feature $x \in H_p(K_i)$ dies at $j$ if $x \not\in H_p(K_j)$. A feature will also die if the feature merges with a feature born earlier in the filtration.

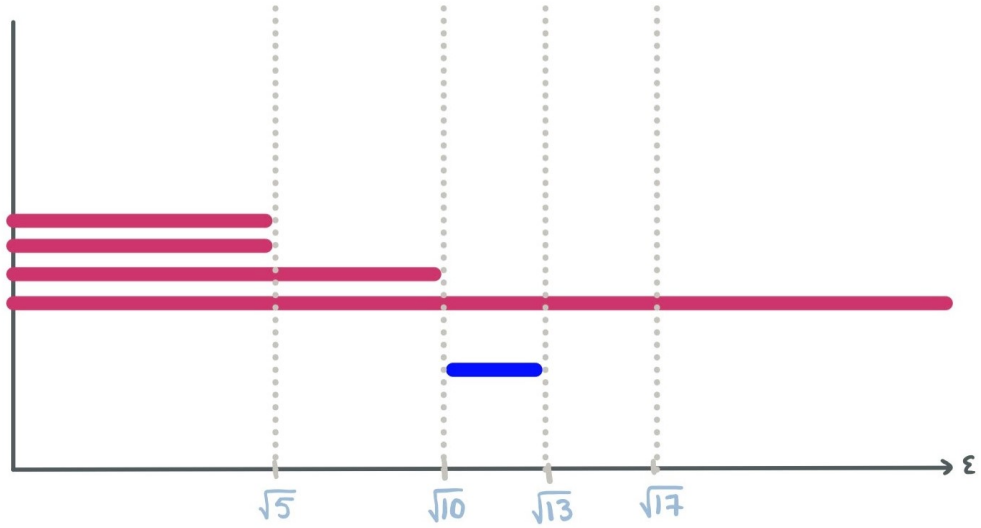
**Definition 3.18** (Persistence interval)**.** For a given topological feature, $x$ with birth point, $i$, and death point, $j$, the persistence interval for the feature is given by $[i, j)$. If $j = \infty$, then the component does not die during the filtration (persists forever).

In this sense, a topological feature is completely characterized by its persistence interval. Graphically, we can represent the persistence intervals of all topological features in a given data set in a couple different ways.

**Example 3.19.**




Visual representation of a Rips filtration for $S = \{(0, 0), (1, 3), (2, -1), (3, 2)\} \subset \mathbb{R}^2$.




Persistence barcodes for the point cloud S = {(0,0), (1,3), (2,-1), (3,2)}

The second image is a persistence bar code that represents the data in the filtration of $S$. At $r = 0$, we have 4 distinct "clusters" of data, and thus we have the beginning of 4 distinct bars leaving the $y$-axis. At $r = \sqrt{5}$, we see that the radii of some of the points overlap, and thus, we see there are now 2 distinct clusters of cells. As a result, 2 of our previous 4 distinct clusters were absorbed, and die at $r = \sqrt{5}$. At $r = \sqrt{10}$, we see that all radii touch, meaning there is only one distinct cluster of cells. We also see that they are joined in such a way to give rise to a new topological feature; a loop is born at $r = \sqrt{10}$. At $r = \sqrt{13}$, this loop dies because an edge breaks the loop in half, creating a higher dimension simplex.

## 3.3 Application to Stem Cells

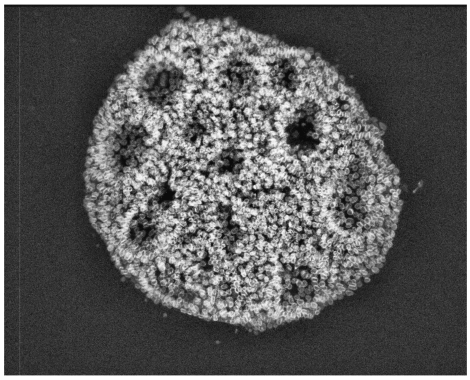
We received images and videos of stem cells from Todd McDevitt, at the time a professor at the Gladstone Institutes at UCSF, in the form of H5 files. In the summer of 2021, I, along with a group of Frost students, extracted point clouds from these files to analyze the persistent homology intrinsic to the data sets.

We used the Ripser Package in Python in order to calculate persistence bar codes from a given data set. These bar codes served as a sufficient comparison in order to identify significant topological features such as loops

and clusters. Further, we were able to take the output and graphically represent our results for an easier comparison between persistence bar codes. [6]

Figure 7: Stem cell cluster with many loop-like structures

(a) Lab-Obtained Image



(b) Persistence Diagram

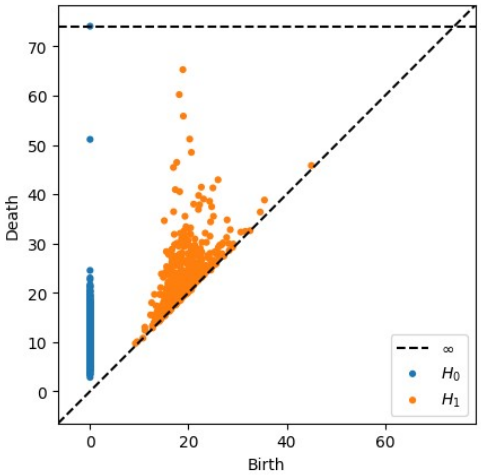
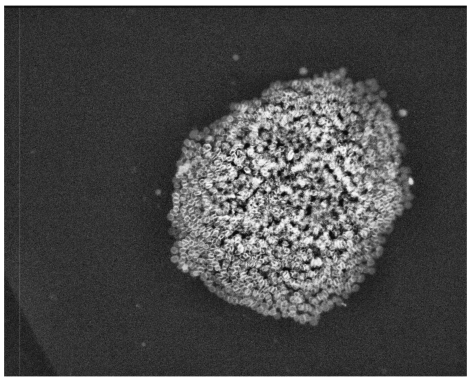
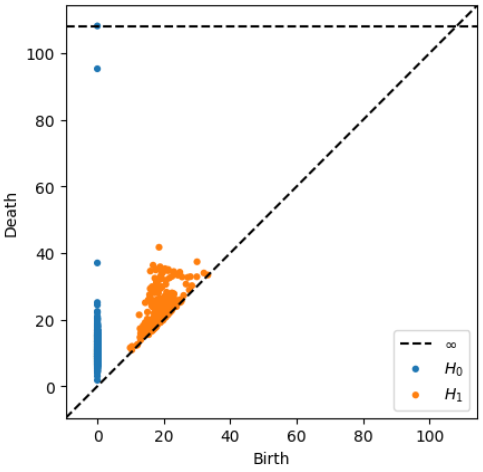



Figure 8: Stem cell cluster with no significant patterning

(a) Lab-Obtained Image



(b) Persistence Diagram



# 4 Agent-Based Modeling

## 4.1 What is an Agent-Based Model?

An agent-based model (ABM) is a computational stimulation of "agents" interacting with each other in a virtually designed environment. An agent is a programmed as an element of a model that can interact with other agents or its environment in a predetermined way. There is a notion of stochasticity encoded into the fabric of an ABM, meant to stimulate our uncertainty in modeling behavior and create a picture more accurate to a lifelike circumstance. The goal of an ABM is to stimulate phenomena that would occur naturally in the world and be able to tweak parameters and conjecture what what would happen if similar action was taken in real life. In these instances, we have more control over our experimentation than we would in a lab setting, making agent-based modelling a very compelling choice for researchers to observe and understand real-world phenomena.

## 4.2 NetLogo

NetLogo was our first tool used developed an agent-based model to stimulate stem cell behavior in-vivo. The purpose of this model was to create a tool for easy analysis of stem cell behavior. Access to real-life images from labs is often limited and finicky due to lack of ideal testing circumstances and the nature of stem cell efficacy. Our goal was to create a model that could output topologically similar point clouds that could be analyzed in tandem with the lab given images. Ultimately, we planned to stimulate the introduction of a variety chemicals and morphogens into the medium to stimulate different behavior and use TDA to characterize the emergent patterning that resulted.

### 4.2.1 Model

**Set-Up:** This model is parameterized by a notion of a time step. At each time interval, each agent present will preform some kind of action and update its stored parameters. Upon instantiation, the model creates a two-dimensional lattice and randomly distributes all agents within designated ranges from the center point.

**Parameters:** This model has four agents: Stem Cells and Morphogens: (Generic Morphogens, BMP4, NOG). The number, size, shape, and color

are all controlled and set to a certain value. Sliders exists outside of the model window to control specified parameters within the code that are used to model behavior. Certain observed quatities are displayed monitors outside of the model.

**Stem Cells:** Each stem cell is encoded to move towards generic morphogens (in an attempt to display the behavior of creating pockets) as well as absorb and keep tabs on the amount of contact they have with BMP4 and NOG agents. After a certain number of time steps, cells will differentiate, and cell fate is displayed via a change of color. At this point, the model stops and we obtain an image that we may analyze via TDA. They are also programmed to reproduced, given they reach a certain energy value, and upon reproduction, split their energy in half to share with their daughter cell. Stem cells are black circles with radius equal to 1. Upon differentiation they will turn red, green, or blue (corresponding to enctoderm, mesoderm, and endoderm respectively).

**Generic Morphogens:** Each morphogen is represented as an orange circle with radius $\frac{1}{2}$ They have no behavior themselves other than to be motionless. All behavior regarding them is implemented through encoded stem cell behavior.

**BMP4:** This morphogen is represented as a blue circle of radius $\frac{1}{2}$ and is encoded to move in the medium at complete random. When it encounters a NOG agent, this molecule will freeze in its position and stop moving for a desginated number of time steps. Its efficacy in adding to a stem cell's chemical contact is also reduced during this time. This is the implementation of the reaction-diffusion.

**NOG:** This morphogen is represented as a purple circle of radius $\frac{1}{2}$ and is encoded to move in the medium at complete random. It moves in farther strides than the BMP4 does in order to show that BMP4 has a much smaller diffusitivity than NOG. Therefore, NOG can be seen in all areas of the model whereas BMP4 cannot.

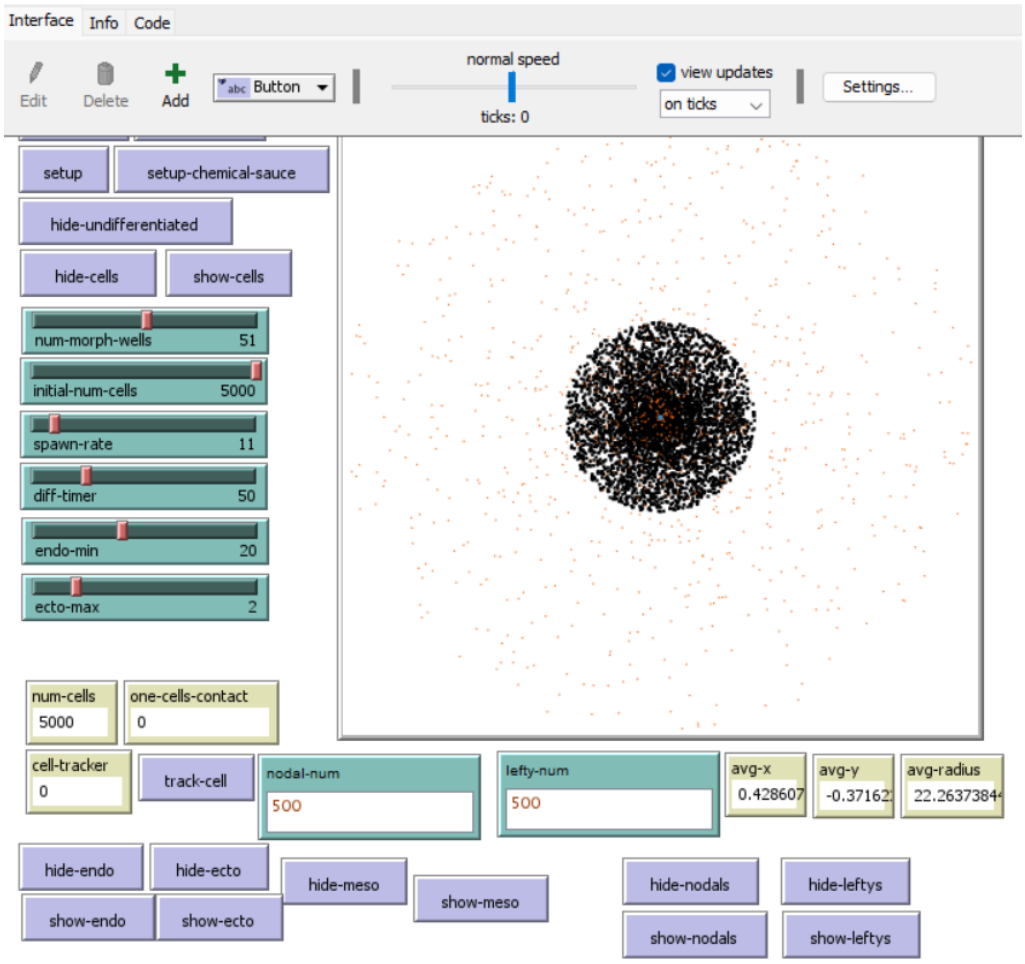



Figure 9: User-Interface of NetLogo ABM

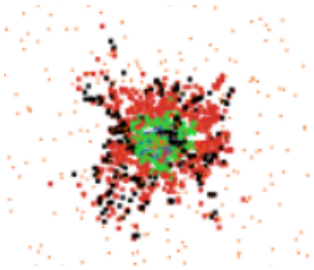


Figure 10: Stimulated Differentiation Gradient from NetLogo Model

### 4.2.2 Code

In this section, we can see some snippets of code that implement some basic functionalities of the model

```
to setup
  clear-all
  ask patches[
    set pcolor white
  ]
  set avg-radius 0
  diff-morph
  create-cells initial-num-cells [setxy 0 0
  set shape "dot"
  set color 0
  set size 1
  lt random 360
  forward random-float 30
  set energy 0
  ]
  set num-cells initial-num-cells
  calc-avgs
  set sauce False
  set tracking 0
  reset-ticks
end
```

Figure 11: NetLogo Set-Up Function

Whenever the model is reset, this set-up function will clear everything being displayed. Then it will initializa and radially distribute cells and generic morphogens into the lattice.

```
to go
  spawn-cells
  calc-avgs
  move-cells
  if sauce = True[
    move-nodals
    move-leftys
    regulating-reaction
    differentiation-tick
    if start-diff = True[
      cascade
    ]
  ]
```

Figure 12: NetLogo Go Function

The go function is the action that the model takes when one time step is tabulated. Every function written within the go function will run once per time step.

```
to spawn-cells
  ask cells[
    if energy >= spawn-rate[
      ask patch-here[
        sprout-cells 1 [
          set color 0
          set shape "dot"
          set size 1
          set energy floor spawn-rate / 2
          set num-cells num-cells + 1
          if sauce = True[
            set differentiated "virgin"
            set time-for-diff diff-timer
            set chemical-contact 0
          ]
        ]
      ]
      set energy floor spawn-rate / 2]
  ]
end
```

Figure 13: NetLogo Spawning Cells Function

When cells accrue a predetermined "spawn-rate" value of energy, they will spontaneously sprout a daughter cell with half of its energy.

```
to differentiation-tick
  ask cells[
    ifelse time-for-diff > 0 [
      set time-for-diff time-for-diff - 1
      let num-nodal 0
      ask patch-here
        [
          set num-nodal count nodals-here with [active = True]
        ]
      if num-nodal > 0[
        set chemical-contact (chemical-contact + num-nodal)]
        ]
    [
      if differentiated = "virgin"[
        set start-diff True
        if chemical-contact >= endo-min [
          set differentiated "endo"
          set color 105
          ]
        if chemical-contact < endo-min and chemical-contact >= ecto-max[
          set differentiated "meso"
          set color 65
          ]
        if chemical-contact < ecto-max[
          set differentiated "ecto"
          set color 15
          ]
      ]
    ]

  ]
end
```

Figure 14: NetLogo Differntiation Monitor

**Note:** In this model, Nodal was the name used for what we now consider BMP4. This algorithm will tabulate the concentration of nodal at each given "patch" in the medium and "expose" the cells to it. After a certain amount of time, cells will differentiate based on the amount of chemical contact they have accrued and change color as a result.

### 4.2.3 Shortcomings

The biggest problem with this model was the lack of an ability to do

collision detection. NetLogo handles location by distinguishing specified discrete patches, which are much larger than the stem cells in the model. In other words, in order to find neighbors within a certain distance, one must utilize the patch native functions to look for nearest occupying patches. This leads to an inaccurate distinction of neighboring cells and does not account for the actual physical contact of cell bodies. In order to accurately deduce when cells intersect and overlap, we either drastically restrict motion (hundreds of cells becomes neighbors) or we ignore the notion of collision and have cells converging to morphogens so quickly that they morph into one point. In order to produce biologically accurate images, this model would be insufficient.

## 4.3 Mesa and Python

After realizing the limitation to the capabilities of the NetLogo version, Python became a more comfortable environment to work in. Instead of a discrete space of patches, this model implements a "Continuous Space" where locations in the medium are represented as arbitrary point values. Initially, the library was relatively incomplete, so new ad-hoc visualization elements needed to be created in order to get the model off the ground. But once this hurdle was overcome, the Python model could exhibit much of the same behavior that was originally implemented in the NetLogo model as well as optimizing and strategizing further improvements.

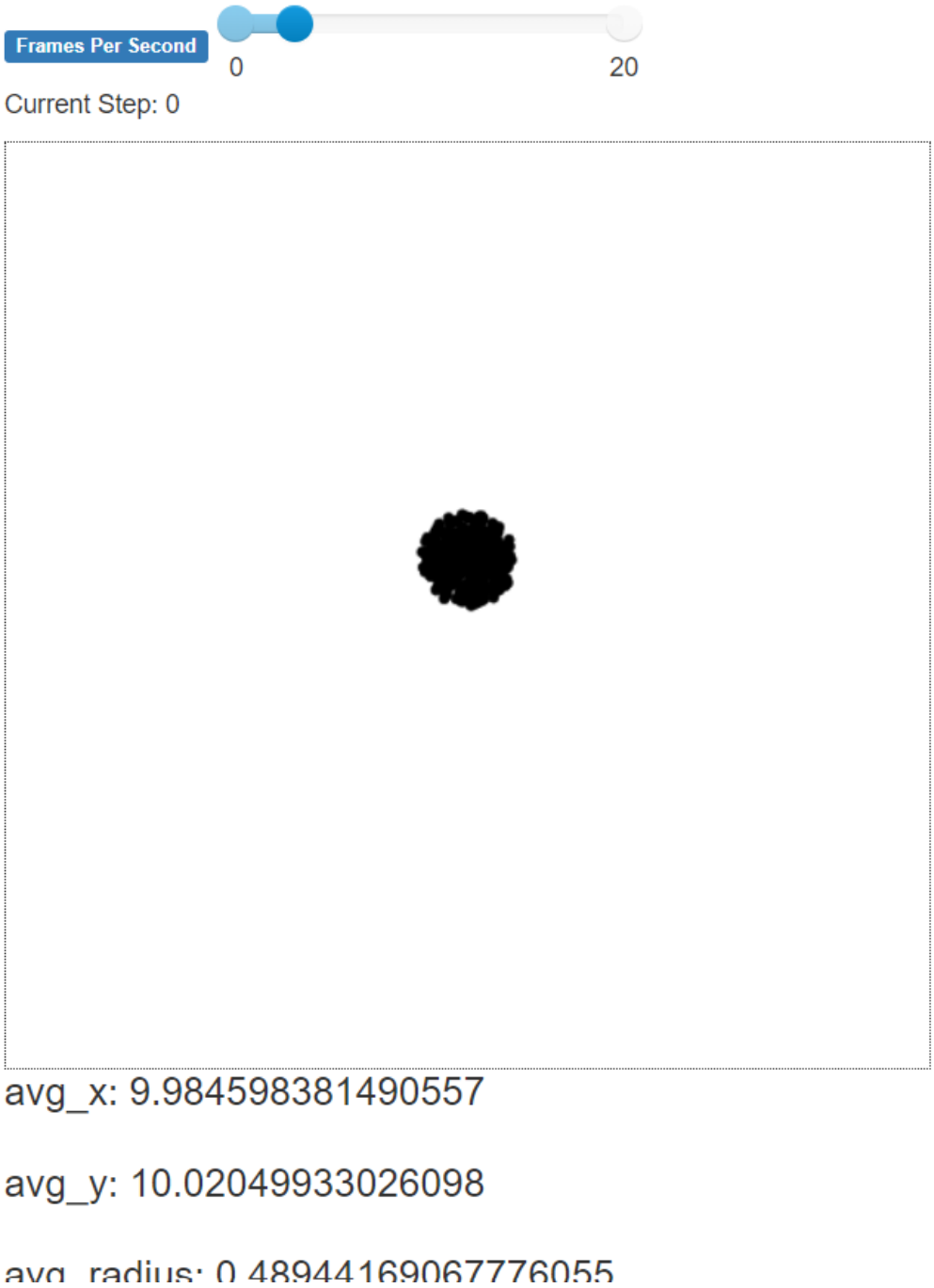



Figure 15: User-Interface of Python ABM

The mesa model is coded in Object-Oriented Programming style: Each agent is represented as a custom data structure. The model data type will take in all the parameters and instantiate a scheduler, a continuous two-dimensional lattice, all of the agents, and steps through all of the agents behaviors at each time interval. A visualization is displayed of the model playing each time step utilizing JavaScript and HTML5 to become compatible with a web browser. Parameters are displayed below the model and updated similar to monitors in the previous model.

GitHub Repository

### 4.3.1 Collision Detection

To put this my algorithm into context, each stem cell will run this algorithm once every time step and has access to all information available native to the stem cell in the medium, including a list of neighboring cells.

Given $c_1$ and $c_2$ be a stem cells, let $r$ be the internal radii for theses stem cells (they all have the same size in this model) and let $(x_1, y_1)$ and $(x_2, y_2)$ be the position of the center of each of these stem cells. Then $c_1$ and $c_2$ are neighbors if $d\big((x_1, y_1), (x_2, y_2)\big) \leq 2r$ where $d(x, y)$ is Euclidean distance. In other words, they must at the very least have touching boundaries.

The main motivation behind my algorithm is to notice the geometric implication of intersecting boundaries. Given two circles overlap, there must be at most two angles with respect to the center of each circle that denote points of intersection on the boundaries of both circles. In a simpler case, where circles intersect at one point, there is only one angle for each circle. In a case where there are no points of intersection, there are no angles to consider. These angles can be interpreted in the standard counterclockwise fashion, considering each circle to be a unit circle.

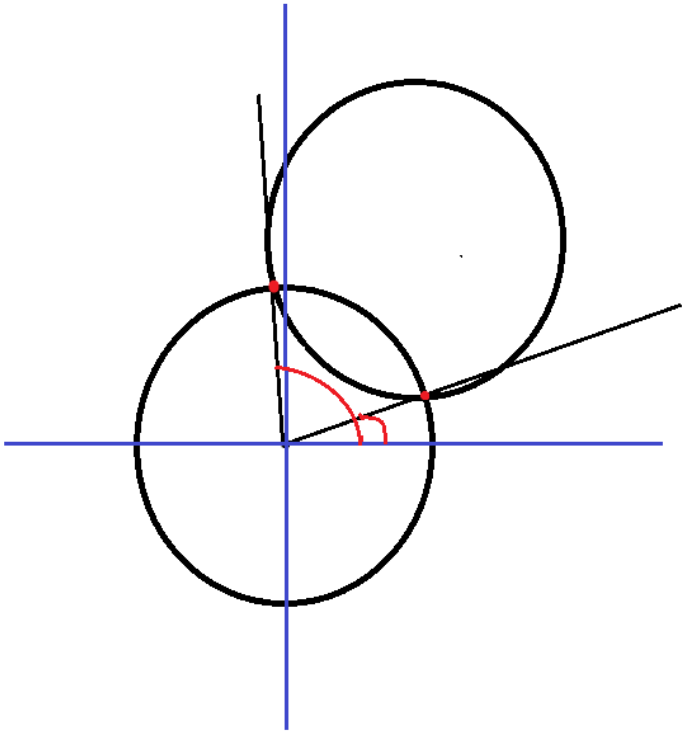


Figure 16: Representation of intersecting cells having two distinct angles along respective perimeters.

In order to determine when cells are overlapping, the movement algorithm was developed around a custom data structure. The SetRange data structure stores as an argument the two angles of intersection corresponding a neighboring cell and acts like a closed numerical range. In other words, SetRange($\alpha$,$\beta$) $\cong [\alpha, \beta]$. However, there is an extra parameter that will help in the event that an intersection occurs that encaptures $\theta = 2\pi = 0$. Clearly, an interval like $[\frac{11\pi}{6}, \frac{\pi}{3}]$ does not make sense. But this data structure is encoded to account for the real meaning: SetRange($\frac{11\pi}{6}$,$\frac{\pi}{3}$) $\cong [\frac{11\pi}{6}, 2\pi] \cup [0, \frac{\pi}{3}]$.

After we deduce the set ranges for all of the neighbors, Another custom data structure combines them into one set of intervals, a SetRangeUnion. Takes all intervals and creates a simplified union of them to be stored as a component for determining where stem cells are not allowed to move. This data structure is designed to reduce and simplify unions to avoid a computationally expensive checking algorithm.

This algorithm will now pick a random direction to move and then

project that motion onto the most feasible path it can take. If the most direct path is not blocked, then it will move straight in that direction. If the most direct path is blocked, then we want to project onto the nearest possible path to move graze past a cell that is blocking the path. This path will intuitively be corresponding to one of the angles of intersection and therefore the angle corresponding to that intersecting point. Finding the minimum difference between angles will give us the angle of the path we want to project onto.

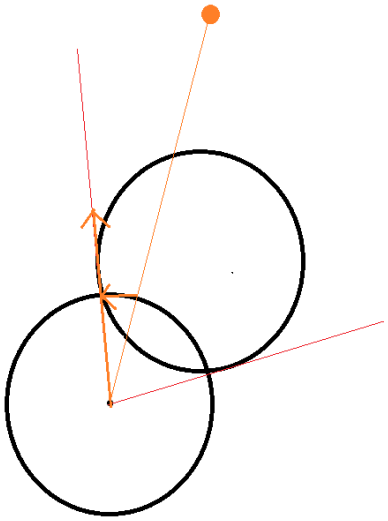


Figure 17: Representation of movement algorithm projecting path onto the closest possible alternative.

Finally, to make the motion more biologically accurate, we can make the speed of the cell moving dependent on how much overlap between its boundaries and its neighbors. That is, the range of $\theta \in [0, 2\pi]$ that is taken up by a SetRangeUnion influences the amount a cell can move. In other words, let $s$ be the range of a given SetRangeUnion for a stem cell. Then that stem cell will move with speed $\frac{2\pi - s}{2\pi}$

### 4.3.2 Modeling Morphogens

In this portion of my senior project, I joined with a group of students and faculty to elevate the model. Specifically, I collaborated with graduate student Brandon Le, in order to develop two different ways to implement

diffusion of morphogens in the model. The following section was written in a collaborative effort between the two of us.

**Reaction-Diffusion**

We wish to model the reaction diffusion equation:

$$u_t = a(u_{xx} + u_{yy}) + g(u) \quad 0 \leq x, y \leq W,\ t > 0$$

where $u(x, y, t)$ describes the concentration of particles, such as BMP4 morphogens, across a square region and time, and the reaction component $g(u)$ describes how the particles react at a certain concentration. Assume zero flux. While there are well-known solutions to the reaction equation, introducing $g(u)$ can make it significantly harder, or even impossible to provide a solution. Thus we turn to finite difference methods, replacing each term in our PDE by a linear formula, and using the tools of linear algebra to provide an approximate solution.

There are many different formulas that can used to approximate derivatives. We'll be using Crank-Nicolson which can be shown to be numerically stable for reaction-diffusion:

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{h} = a\Big(\frac{L_x u_{ij}^{n} + L_x u_{ij}^{n+1}}{2} + \frac{L_y u_{ij}^{n} + L_y u_{ij}^{n+1}}{2}\Big) + g(u_{ij}^{n})$$

where $u_{ij}^{n}$ is position $i, j$ at time $n$ of our finite mesh discretization (we break apart our region into small squares of length $k$), $h$ is the time-step and $L_{x,y}$ is the finite difference approximation of the second derivative with respect to $x$ and $y$, respectively. We'll use $u''(x) \approx \dfrac{u(x+k) - 2u(x) + u(x-k)}{k^2}$ as the 2nd derivative approximation.

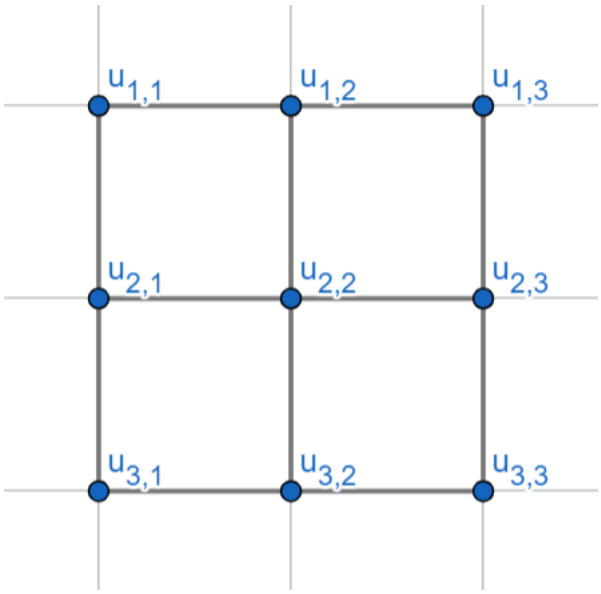



Figure 18: The unit square, broken up into distinct, evenly-spaced points

**Example 4.1.** If we break the unit square in Figure 18 into this finite scheme (just 9 points as a discrete representation of the square), we see that for $L = L_x + L_y$,

$$Lu^n_{2,2} = 4(u^n_{2,1} - 2u^n_{2,2} + u^n_{2,3}) + 4(u^n_{3,2} - 2u^n_{2,2} + u^n_{1,2})$$

Notice how we have an equation for each grid point $u_{ij}$ in our domain, or a system of linear equations. Moving the future time-steps ($u^{n+1}$) to the LHS and the present time-steps ($u^n$) to the RHS, we get:

$$(\frac{1}{h}I - \frac{aL}{2})\vec{u}^{n+1} = (\frac{1}{h}I + \frac{aL}{2})\vec{u}^n + g(\vec{u}^n),$$

where $I$ is the identity matrix, and

$$\vec{u}^n = \begin{bmatrix} u_{11}^n \\ u_{12}^n \\ \vdots \\ u_{1n}^n \\ u_{21}^n \\ \vdots \\ u_{2n}^n \\ u_{31}^n \\ \vdots \\ u_{mn}^n \end{bmatrix}$$

This allows us to solve for $\vec{u}^{n+1}$, the concentration values at the next time step, by inverting the LHS matrix.

**Brownian Motion**

Brownian motion is a stochastic process used to model the random motion of very small particles. It is a continuous process, but we can approximate Brownian motion by displacing each particle by a distance following either a random walk or a normal distribution along each time-step. For example, if $B(0) \sim \vec{x}$ describes the position of one particle at time 0, position $\vec{x}$, then if we displace it by a normal distribution with variance $h$ then $B(1) \sim B(0) + N(0, \sqrt{h})$, $B(2) \sim B(1) + N(0, \sqrt{h})$, and so on.

**Remark:** Moving by a random walk, normal distribution, or some other random variable means we pick an event, which has some probability to occur out of all possible events, and then move by that event. So for a random walk in 1-dimension, there is a 50% chance the particle will either move left or right by a flat amount, while for a normal distribution the particle will mainly move within 2 standard deviations from it's last position.

Unfortunately, the mathematical definition of Brownian motion doesn't fully align with the physical phenomenon of Brownian motion. We need to account for collisions of particles as they're integral to detect cell differentiation. To do so, we construct a distance matrix of all the particles, where the $(i, j)$th entry returns the distance between particle $i$ and particle $j$. When two different particles are lower than some distance threshold, which depends on the types of particles themselves, we make the particles "bounce" off, moving in the opposite direction with some random variation.

For the project, this model was hard-coded so that when BMP4 and NOG collide, only BMP4 bounces off, as NOG inhibits BMP4. Stem cells don't react with collisions.

### 4.3.3 Code

In this section we see some snippets of code that implement some important features of the Python model.

```
def setup(self):
    #Add StemCells to the Space
    if self.hasCells == True:
        for i in range(self.num_stem_cells):
            self.currentIDNum += 1
            c = StemCell(self.currentIDNum , self)
            self.schedule.add(c)
            r = self.random.random() * 1
            theta = self.random.random() * 2 * math.pi
            x = r * math.cos(theta) + self.center_pos[0]
            y = r * math.sin(theta) + self.center_pos[1]
            self.space.place_agent(c , (x , y))
            self.cells.append(c)
        self.stem_cell_ex = self.space._index_to_agent[self.random.randrange(0 , len(self.space._agent_points))]
        self.stem_cell_ex_diff = self.stem_cell_ex.differentiated
```

Figure 19: Python Set-Up Function

This is the Python model's set up function. It functions almost identically to the NetLogo version, although it can be noticed that certain features of the Mesa Library, like adding agents to a "schedule" are explicitly necessary for expected functionality.

```
def step(self):
    self.calcAvgs()
    if self.hasCells:
        self.updateParams()
        self.updateTouchingDict()
        self.updateBMP4()
    if self.start_diff == True:
        self.cascade()
        if self.end_time == -2:
            self.end_time = 3
        self.end_time -= 1
    self.schedule.step()
    if self.end_time == -1 and self.start_diff == True:
        self.running = False
```

Figure 20: Python Step Function

This is the Python model's "go" function. It is very similar to the NetLogo version. A notable difference is the inclusion of a "updateTouchDict()" which keeps track of stem cell neighbors.

```
class SetRange:

    def __init__(self , num1 , num2 , closed , lapped):
        self.lower = min(num1 , num2)
        self.upper = max(num1 , num2)
        self.closed = closed
        self.lapped = lapped
```

Figure 21: Custom Data Structure: SetRange

This is the initialization function for our custom data structure, SetRange. The "lapped" argument determines whether the interval contains 0 or $2\pi$.

```
zpRange = SetRange(zpAngle + math.pi / 2 , zpAngle - math.pi / 2 , True , 1)
noThetas = True
for theta in thetas:
    if zpRange.numInRange(theta):
        noThetas = False
if not noThetas:
    if neighborSet.numInRange(zpAngle):
        zpAngle = self.getMinDistanceAlongCircumference(zpAngle , thetas)
x = self.internalR * math.cos(zpAngle)
y = self.internalR * math.sin(zpAngle)
head1 = self.model.space.get_heading(self.pos , (self.pos[0] + x, self.pos[1] + y))
head1 = (head1[0] *(((2*math.pi)-r)/(2*math.pi)) , head1[1]*(((2*math.pi)-r)/(2*math.pi)))
h = head1[0]
v = head1[1]
norm = scaleFactor * (h ** 2 + v ** 2) ** (0.5)
if h == 0 and v == 0:
    norm = 1
newPos = (self.pos[0] + (h/norm) , self.pos[1] + (v/norm))
self.model.space.move_agent(self , newPos)
```

Figure 22: A snippet from the movement algorithm

This is a piece of our movement function. In this portion, we check within a range of $\pi$ in the direction facing our desired movement whether or not movement is blocked by any of our angles given by overlapping cells. If there is blocking, we get the angle corresponding to the minimum distance and move in that direction. Otherwise, we move in the original direction.

# 5 Closing Remarks

## 5.1 Future Work

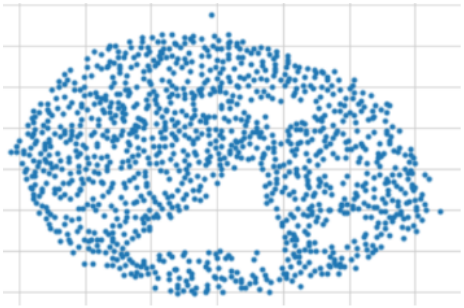


Figure 23: Point Cloud extracted from Lab Image

Figure 24: Point Cloud generated by Python Model

As shown in Figures 23 and 24, our model does not have the capability yet to produce visually similar point cloud images. To create an output more accurate to a lab induced scenario, we need to identify just how powerful the morphogen influence is on overall cell motion. If BMP4 has too strong a pull in the model, we will see cells break their clustering structure and behave abnormally. But on the other hand, if the BMP4 force exerted is too weak, we will see cell-on-cell adhesion forces collapse the cluster in on itself even as new cells get produced. A balance must be fine-tuned for this model to become life-like.

While the diffusion algorithms exist, they still need to be integrated into the ABM itself. Accurately modelling the reaction-diffusion of morphogens in the medium is key to the development of topological features as well as the germ layer differentiation gradient. Combatting issues like run-time efficiency is the biggest challenge, as we are working with many tiny particles. It may be more useful to hard code and import some unchanging data that we can acquire from assumptions made about the initial and boundary conditions.

After these two adjustments are made, the model should become more realistic, and we should be able to do TDA on point clouds of cell clusters that we extract. We know we have created a life-like model if our Ripser analysis can give us similar persistence bar codes to those of similar looking point clouds generated from a lab image. This marker will tell us that behavior seen in cells can be effectively modeled and compared across real life and stimulated images. Further, this ABM becomes an ideal test-bed for other chemical phenomena that could alter the course of gastrulation.

Studying the process of embryonic development is a very difficult task in a lab setting; behavior of stem cells are rather finicky making it difficult to effectively test hypotheses. This ABM circumvents the barrier created by temperamental stem cell behavior. Further, it supports the potential of added functionality to test new chemicals and morphogens in our stimulated lattice. It will become a cost-effective and consistent means to analyze stem cell movement and ultimately differentiated germ layer fate. What makes this model so powerful is the use of TDA to confirm that images taken from the model match closely with those from already known studies and lab research. With this quantitative analysis, our model will show that the implementation of new morphogens and chemical processes are realistic. This model becomes our insight into patterns that the human eye cannot detect just by looking through a microscope. Giving the seeming chaos of nature a well structured code representation is our tool to understand the

principles that govern this complex process.

## 5.2 Conclusion

With the combined power of TDA and agent-based modeling, we can characterize patterns in stem cell clusters that are not necessarily apparent to the human eye. With classification of different homologies, we can make comparison and conjectures about images of stem cells from different labs as well as models, ultimately finding the essential similarities that govern stem cell motion. While this project's scope was limited to the examining and analyzing of only stem cell behavior, the broader application of TDA in other areas of biology cannot be undervalued. Wherever patterns exist, TDA is a readily available tool that can characterize the most important geometric features and streamline the process of comparing and contrasting.

There is undoubtedly still work to be done to make this ABM a more accurate depiction of the gastrulation process. Optimizing diffusion and cell differentiation algorithms will serve to stimulate the realism we seek for comparison. TDA will give us the closure we need to accurately claim that this model stimulates real biological phenomena. With these tools used in conjunction, we can then elevate our design to implement new biological processes that work in tandem with our stimulated embryonic development. We can further test the effectiveness of other stimulated chemicals to influence the model and whether their impact is life-like. All of these tools have created a test-bed for any inter-cellular communication and behavior to be modeled and compared with real world behavior. With this, we can test hypotheses, make discoveries, and more efficiently make contributions to the stem cell community than ever before.

# References


[1] Vazin, Tandis, and William J Freed. "Human embryonic stem cells: derivation, culture, and differentiation: a review." Restorative neurology and neuroscience vol. 28,4 (2010): 589-603. DOI: doi.org/10.3233/RNN-2010-0543

[2] Gilbert SF. Developmental Biology. 6th edition. Sunderland (MA): Sinauer Associates; 2000. Endoderm. URL: https://www.ncbi.nlm.nih.gov/books/NBK9983/

[3] Ansari A, Pillarisetty LS. Embryology, Ectoderm. [Updated 2022 May 8]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2022 Jan- URL: https://www.ncbi.nlm.nih.gov/books/NBK539836/

[4] Mukul Tewary, Joel Ostblom, Laura Prochazka, Teresa Zulueta-Coarasa, Nika Shakiba, Rodrigo Fernandez-Gonzalez, Peter W. Zandstra; A stepwise model of reaction-diffusion and positional information governs self-organized human peri-gastrulation-like patterning. Development 1 December 2017; 144 (23): 4298–4312. DOI: https://doi.org/10.1242/dev.149658

[5] Müller P, Rogers KW, Yu SR, Brand M, Schier AF. Morphogen transport. Development. 2013;140(8):1621-1638. DOI: https://doi.org/10.1242/dev.083519

[6] Tralie et al., (2018). Ripser.py: A Lean Persistent Homology Library for Python. Journal of Open Source Software, 3(29), 925, DOI: https://doi.org/10.21105/joss.00925

[7] Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[8] Kazil, Jackie et al., Social, Cultural, and Behavioral Modeling, Utilizing Python for Agent-Based Modeling: The Mesa Framework. Springer International Publishing. 2020 URL: https://github.com/projectmesa/mesa