Modeling Vascular Diffusion of Oxygen in Breast Cancer

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Tina Giorgadze

Annandale-on-Hudson, New York
May, 2023

Abstract

Oxygen is a vital nutrient necessary for tumor cells to survive and proliferate. Oxygen is diffused from our blood vessels into the tissue, where it is consumed by our cells. This process can be modeled by partial differential equations with sinks and sources. This project focuses on adding an oxygen diffusion module to an existing 3D agent-based model of breast cancer developed in Dr. Norton's lab. The mathematical diffusion module added to an existing agent-based model (ABM) includes deriving the 1-dimensional and multi-dimensional diffusion equations, implementing 2D and 3D oxygen diffusion models into the ABM, and numerically evaluating those equations using the Finite Difference Method. I started by diffusing a point source in a 2D grid, then diffusing a line in a 2D grid, then a cubic patch in a 3D grid, and finally, diffusing oxygen from blood vessels into the tissue. Then, I programmed the supply function to represent the continuous oxygen supply from vasculature, and the uptake function to represent the oxygen uptake by cancer cells.

Contents

Dedication

This project is dedicated to my late grandmother Tina, who passed away due to breast cancer in 2017. She made me the person I am today.

Acknowledgments

I want to thank my advisors Dr. Kerri-Ann Norton, Ethan Bloch, and Sven Anderson for supporting me throughout my senior year. Additionally, I am grateful to every professor in the computer science and mathematics departments for providing me with the tools and skills necessary to successfully complete my senior project. Last but not least, I want to acknowledge my colleagues Ansel Tessier and Henning Fischel, who created a supportive and collaborative space for me during Bard Summer Research Institute 2021, when I discovered my deep interest and passion for computational biology and cancer modeling.

1
Introduction

The purpose of this project is to add an oxygen diffusion module to an existing agent-based model of triple-negative breast cancer developed in Dr. Norton's lab in order to measure hypoxia in cancer more accurately.

Cancer is identified by the uncontrollable expansion of cells in our bodies, which includes several hallmark abilities such as unlimited replicative ability, ability to invade and spread, suppressing growth suppressors, and more ([20]). This makes tumors difficult to treat, with frequent remissions and recurrences. Triple Negative Breast Cancer (TNBC) is a particularly difficult tumor to treat, as it lacks the three most commonly targeted receptors ([13]). Dr.Kerri-Ann Norton's computational biology lab has developed a 3-dimensional agent-based model of TNBC. The most recent research purpose of Norton's lab was to simulate the effects of a newly developed immunotherapy on the tumor and observe the parameters and conditions that affect the effectiveness of this treatment. ([11], [16])

Diffusion refers to the movement of material from high to low concentrations ([1]). A classic example of this process is the movement of heat through a rod. This project looks at the diffusion of oxygen from our blood vessels into the tissue. Just like heat flows from warmer to colder temperatures, oxygen diffuses from higher to lower concentrations. Once the oxygen is diffused into the tissue, it is then absorbed by our cells as a form of nutrition necessary to

support cell life and reproduction. Like any other cells, cancer cells rely on oxygen to sustain their lives and expand ([12]). Hypoxia refers to cells being too far from the vasculature to obtain oxygen, and cells in hypoxic conditions have different properties. For example, hypoxic cells have slower proliferation and migration rates than cells that have access to oxygen and nutrition ([28]). The TNBC model this project builds upon incorporates hypoxia and alters tumor behavior accordingly. Therefore, having an accurate and realistic representation of the movement of oxygen molecules through our tissue and to the cancer cells is crucial for having our model yield realistic results.

This project aims to improve the model by adding an oxygen diffusion module, which includes several steps. First, the design of accurate partial differential equations to model diffusion with sources (vasculature) and sinks (tumor cells absorbing oxygen). Second, a numerical evaluation method will be necessary to estimate the value (amount) of oxygen at a given location. This method will have to apply to three spatial and one time dimension, as the existing method is 3-dimensional. Third, the mathematical model, which is sometimes referred to as a continuous model, will have to be integrated into the existing agent-based model. After the model is complete, it will be validated to prove its correctness.

1.1 Biology Background

1.1.1 Cancer

Cancer manifests itself in many forms, all of which have different complex properties and behaviors. There are several steps to tumor development and progression. Zijl et al. ([59]) explain the process of tumor development: the first step in the formation of tumor is the mutation of cells. These mutated cells have an increased risk of becoming cancerous. This process is known as the initiation. The cells that undergo such genetic mutations are then subject to the second stage of cancer development known as the promotion, where the proliferation rate of such cells starts to change. The next stage, the progression, is when the growth rate of the cells increases, and the cells undergo more mutations that increase their invasiveness. As seen in Siddiqui et al.

([51]), the invasiveness of cancer implies that it has the ability to spread from one location to another through either the lymph system or the bloodstream. Therefore, cancer cells need to enter the blood vessels, survive inside the bloodstream, leave the vessel, and then keep growing and proliferating at the new location away from the primary site.

Hanahan and Weinberg ([20]) have identified eight capabilities that most solid tumors share, including triple-negative breast cancer - the cancer simulated in this project. These eight hallmarks that make tumor incredibly difficult to treat include sustaining proliferative signaling, evading growth suppressors, resisting cell death, enabling replicative immortality, inducing angiogenesis, activating invasion and metastasis, deregulating cellular energetics and metabolism, and avoiding immune destruction. Since these hallmark capabilities contribute to the growth and development of cancer, producing drugs to inhibit these capabilities has been an important task in developing effective therapeutics. Most of the existing cancer drugs target specific hallmark capabilities. However, many of these drugs have been observed to have temporary effects, with the tumor eventually developing resistance or going into relapse ([20], [57]).

There are several reasons why this happens. First, it has been observed that hallmark capabilities are regulated by multiple pathways, not just one. Therefore, when a given drug targets a specific pathway, signals can still travel through other pathways not targeted by the mechanism. This allows some cancer cells to survive, adapt and develop resistance leading to continued tumor growth ([30]). One possible solution to this challenge would be designing a cancer drug that targets multiple pathways simultaneously. That, in theory, would disable the tumor-promoting hallmark capability altogether ([20]). Another reason current drugs can be ineffective in the long run is the ability of cancer to shift its dependence. When one capability is attacked or disabled, the tumor has been observed to become more dependent on another hallmark capability. This means that the tumor can still progress despite the therapy ([20]). A solution to this challenge could be to use a single drug that targets most if not all, hallmarks of tumors. Creating a drug like this is an extremely difficult task, as most processes utilized by cancer cells are also used by the rest of our cells. Therefore, disabling these processes would

cause the patient's body to stop functioning properly and possibly endanger their lives even further. Therefore, a therapeutic mechanism will need to carefully select what processes are non-essential for human bodies that we can risk eliminating.

As described in ([18]), another big reason tumors are difficult to treat is the fact that it lives in a microenvironment that is altered to support tumor growth. This microenvironment can include invasive cancer cells, cancer stem cells, inflammatory immune cells, endothelial cells, cancer-associated fibroblasts, blood vessels, etc. All these cells are normally present in non-cancerous tissue as well, but they all serve distinct functions in the context of the tumor microenvironment. For example, blood vessels support tumor growth by providing oxygen and nutrients. Cancer-associated fibroblasts have been observed to provide structural support. Inflammatory immune cells, or IICs, support tumor progression by increasing cells' proliferation rates as well as death resistance. Tumor cells that have been deprived of oxygen are called "hypoxic" cells. Such cells have reduced migration and proliferation rates. Because of the adverse effects hypoxia has on tumors, treatments that focus on blocking angiogenic signaling can halt new vessel growth, and thus keep tumor dormant. Out of the eight hallmark abilities mentioned above, the one most relevant to this project is the ability of tumors to induce angiogenesis. Since we aim to model diffusion of oxygen from vasculature in TNBC, it is important to understand how cancer affects our blood vessels. In the next section, I discuss angiogenesis, the recruitment process of blood vessels by cancer, as well as its effects on tumor growth.

1.1.2 Angiogenesis

Like any other living cell, tumor cells require nutrients to survive and reproduce. Most essential nutrients and oxygen necessary for tumor cells to live and proliferate come from blood vessels, also known as vasculature ([28]). As discussed above, one of the major hallmarks of cancer includes the ability to recruit new blood vessels. That way, if the tumor undergoes metastasis, it can maintain a supply of oxygen and nutrients wherever it migrates. The recruitment of new blood vessels is known as angiogenesis. In order for a tumor to recruit new blood vessels, it

needs to send a signal that triggers the formation of new blood vessels. This process is known as the "angiogenic switch".

The process of new capillaries branching from existing vasculature is known as sprouting angiogenesis. As discussed in [31], there are three main types of cells that participate in this process: tip, stalk, and phalanx cells. First, the tumor needs to send a signal to endothelial cells to begin sprouting. A major example of such signaling is Vascular Endothelial Growth Factor, or, VEGF, that cancer cells secrete. Endothelial cells are cells that make up the inner lining of blood vessels. They also serve as an anticoagulant barrier between blood vessels and the blood. Once this signal is received, an endothelial cell starts to sprout, or, migrate away from the parent vasculature. The guiding cell is known as a tip cell. This cell guides the sprouting of a capillary toward the signal, usually in the direction of the tumor. Once a tip cell starts to migrate, other endothelial cells (stalk cells) behind it start to divide and migrate together with the tip cell, which creates a sprout. Finally, phalanx cells are the remaining quiescent (dormant) endothelial cells that have lower proliferation and migration rates. Once the branching is finished, the sprouts connect and form lumen in order for blood to start flowing through the newly sprouted blood vessels. This process is known as anastomosis ([53],[22]). Once the new blood vessels undergo anastomosis, the tumor gains access to oxygen and nutrients, allowing it to survive and grow.

My model uses PDEs to simulate oxygen diffusion from blood vessels into the tissue. An important long-term goal of this project is to compare hypoxia rates between my model and previous models. Unfortunately, it is not easy to observe vasculature growth or its effects on a tumor in patients and in-vivo models. It takes time for the capillaries to grow, and the number of studies that observed this process from images is low ([54]). For these reasons, computational modeling is a commonly used tool to observe and predict the effects and patterns of angiogenesis. It allows us to make these predictions and observations in a fraction of the time. For example, a computational model was used to predict that stalk and tip cell proliferation rate has a stronger effect on vasculature growth than migration rate. Models have also been used to

observe vasculature as a drug delivery into the tumor microenvironment. Computational models also allow us to simulate the secretion and effects of VEGF diffusion from cells, as seen in [39]. This model incorporates an angiogenesis module from a previous paper ([16]) in order to diffuse oxygen from capillaries into the tissue.

1.2 Mathematics Background

This section goes over the necessary mathematical tools to model the diffusion of oxygen from blood vessels into the tissue. First, I derive the standard heat diffusion equation, which, in a slightly adapted form, was used in our model. Since our model is 3-dimensional, I used a 3-dimensional diffusion equation. I will show the derivation of the diffusion equation with one spatial dimension, followed by the diffusion equation for higher dimensions.

Second, I will discuss ways to evaluate the 3-dimensional diffusion equation numerically. The method discussed in this section is called the finite difference method. First, I derive the FDM for 2 spatial dimensions, and then show the formula for 3 dimensions.

1.2.1 Partial Differential Equations (PDEs)

Partial differential equations, or PDEs, are equations that depict a relationship between an unknown function and its partial derivatives. These functions in PDEs, unlike in ordinary differential equations, depend on more than one variable; for example, time and space as opposed to just time or just space. Lots of physical phenomena can be modeled and described through PDEs. Some common examples include the heat equation, the wave equation, laws of motion, certain quantum mechanics phenomena, etc. In other words, many processes that describe the change of a quantity over time across space can be modeled by a PDE. The mathematical model described in this project will implement a type of diffusion equation. Therefore, I begin by exploring the derivation and evaluation of the classic diffusion equation.

1.2.2 The Diffusion Equation With One Spatial Dimension

Diffusion refers to the process of matter moving from high to low concentrations in some medium. A classic example comes from a physics problem that looks at heat diffusing through a rod. I begin with a rod of fixed length L, as shown in Figure 1.2.1. Heat is traveling through this rod from one end to another. Let $U(x,t)$ be a function representing the concentration of heat (or other substance) at time t at location x. In order to be able to calculate the function U at a given time t at a given location x, I need to write and evaluate a partial differential equation. In this chapter, I set up the mathematical model and thus derive the classic diffusion equation. I will closely follow the approach seen in [2]. A more concise derivation of the same result can be found on page 126 of [5].

Before I set up the PDE, I have to specify our *initial* and *boundary* conditions. Initial conditions refer to the temperature throughout the rod at time $t = 0$. If $U(x,t)$ denotes temperature at time t and location x, some examples of initial conditions can include, for example, $U(x,0) = 0$, $U(x,0) = \sin x$, etc. Boundary conditions refer to the temperature of the rod on the boundary. Namely, when $x = 0$ and when $x = L$. For example, I could have

$$\begin{cases} U(0,t) = 0 \\ U(L,t) = T, \end{cases}$$

for some temperature T, and for $0 < t < \infty$.

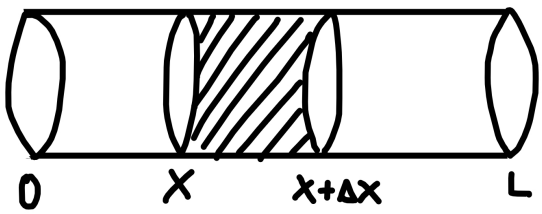

Figure 1.2.1: Thin Rod With Cross Section

Next, I derive the heat equation commonly known as $U_t = U_{xx} + F(x,t)$, where $U_t = \frac{\partial U}{\partial t}$, $U_{xx} = \frac{\partial^2 U}{\partial x^2}$, and $F(x,t)$ is the source/sink function. To keep things simple, I will ignore constant coefficients that would come up in physical occurrences. For this part, I think about how heat moves through the rod over time - in other words, what factors affect U_t. The change in

temperature at a given location is affected by two things: first, the flow of heat through that location. Second, any outside sources or sinks affecting the rod at that location. So, we have

change in U in time = Flow of heat at $x, t +$ external heat from sources or sinks at x, t.

Let $Q(x, t)$ be the heat flow (or flux) function through the rod. In other words, the function represents how heat flows from one location to the other. We define heat to flow from left to right. That means that when heat flux is negative, we have a bigger temperature on the right boundary compared to the left boundary. So, when the flux is negative, we have a positive change in the temperature at that location. Therefore, we need to take the negative of the heat flow in the equation. Then, to show how the heat flow changes at a given location, we take the space derivative of Q. That way, we can quantify the change of heat flow at a given location at a given time. Now, we can rewrite the equation for U_t above to see that

$$U_t = -Q_x + F(x, t).$$

Next, we derive Q in terms of U. A good way to demonstrate the process is to slice our rod into tiny cylindrical pieces starting at x and ending at $x + \Delta x$ as shown in Figure 1.2.1, where Δx is arbitrarily small. Then, we see that the flow of heat through this cylindrical slice of the rod is the difference in the temperatures at the two ends of the slice averaged through the length of the slice. So, to find the flux of heat through this slice, we have $Q_x = \frac{U(x,t)-U(x+\Delta x,t)}{\Delta x}$. However, since we are interested in the instantaneous flux of heat at x as opposed to through the slice, we take the limit as Δx goes to 0. This gives us $Q_x = \lim_{\Delta x \to 0} \frac{U(x,t)-U(x+\Delta x,t)}{\Delta x} = -U_x$. Substituting the above equation in our current diffusion equation, we get

$$U_t = -(-U_x)_x + F(x, t) = U_{xx} + F(x, t).$$

This concludes our derivation of the heat equation with one spatial dimension.

1.2.3 Diffusion Equation For Higher Spatial Dimensions

Let us generalize the diffusion equation for higher dimensions. For this part, we follow the same general structure as in the previous chapter. However, we now focus on a 3-dimensional

volume instead of a one-dimensional rod. This volume can have arbitrary shape and size. This derivation will closely follow the structure seen in [3]. For more detailed derivation of the same result using the same Gaussian theorem we will use below, see Section 1.5 of [19].

Let us once again think about what components affect the total change of temperature in time. As we saw with the rod example, two factors contribute to the total change in temperature. First, heat can be flowing into or from the object. But, since we now have an object with volume, heat would flow through the boundary. Second, we can have an additional source or sink of heat inside the volume (for example, the object can receive an external hit, which can generate additional heat energy). So, we have a rough equation that looks like

Total change in Temperature = Heat flowing through the boundary + External source or sink

First, let us think about how to calculate heat flux through the boundary. Here, we refer to Fourier's Law of Heat Conduction ([36]).

Theorem 1.2.1 (Fourier's Law of Heat Conduction)**.** *The rate of heat transfer through a material is proportional to the negative gradient in the temperature and to the area, at right angles to that gradient, through which the heat flows.*

Putting Fourier's Law in differential form, (ignoring constants), we get $\vec{Q}(X,t) = -\vec{\nabla}U(X,t)$, where X is the spatial position vector and Q represents heat flux. However, recall that we define heat to flow from left to right. Therefore, in our final equation, we need to use the negative of heat flux, because the body gains heat if Q is negative. Additionally, we need to take the dot product of Q with the normal vector $\vec{n}$ pointing perpendicular to the surface. This way, we get the direction of the heat flux to or from the object. Then, if V is the volume of the object, and S is its surface, we need to look at the change in total heat energy of the object as well as the total flux through the entire surface and the total additional heat energy from sinks/sources across the volume. This calls for volume and surface integrals. Therefore, we end up with

$$\frac{\partial \iiint_V U(X,t)\,dV}{\partial t} = -\iint_S \vec{Q}(X,t)\cdot\vec{n}\,dS + \iiint_V F(X,t)\,dV.$$

In order to make matters easier, we need to find a way to convert the surface integral into a volume integral. For that, we refer to Gauss's Theorem of Divergence (see [29]).

Theorem 1.2.2 (Gauss's Theorem of Divergence)**.** *Suppose V is a subset of $\mathbb{R}^3$ (in other words, V represents a volume in three-dimensional space) which is compact and has a piece-wise smooth boundary S (also indicated with $\partial V = S$). If F is a continuously differentiable vector field defined on a neighborhood of V, then*

$$\iiint_V (\nabla \cdot F)\, dV = \iint_S (F \cdot \vec{n})\, dS.$$

The left side is a volume integral over the volume V, the right side is the surface integral over the boundary of the volume V. The closed surface ∂V is oriented by outward-pointing normal ∂V. d *may be used as a shorthand for* dS*.) In terms of the intuitive description above, the left-hand side of the equation represents the total of the sources in the volume V, and the right-hand side represents the total flow across the boundary S.*

Using this theorem, we can rewrite

$$\iint_S \vec{Q}(X,t) \cdot \vec{n}\, dS$$

as

$$\iiint_V (\vec{\nabla} \cdot (Q(X,t)\, dV.$$

Then, our final equation ends up being

$$\frac{\partial \iiint_V U(X,t)\, dV}{\partial t} = -\iiint_V (\vec{\nabla} \cdot (Q(X,t)\, dV + \iiint_V F(X,t)\, dV$$

Next, we use Fourier's Law to write

$$\frac{\partial \iiint_V U(X,t)\, dV}{\partial t} = \iiint_V (\vec{\nabla} \cdot \vec{\nabla} U(X,t)\, dV + \iiint_V F(X,t)\, dV$$

We rearrange the terms to get

$$\iiint_V U(X,t)_t - \vec{\nabla}^2 U(X,t) - F(X,t)\, dV = 0.$$

However, recall that this equation is true for arbitrary volume V. Therefore, the only way for this volume integral to always equal 0 is if the expression inside the integral itself equals 0. Therefore, we get

$$U(X,t)_t - \vec{\nabla}^2 U(X,t) - F(X,t) = 0,$$

therefore,

$$U(X,t)_t = \vec{\nabla}^2 U(X,t) + F(X,t).$$

This equation is known as the higher dimensional heat diffusion equation. We will use this later on in our mathematical model of oxygen diffusion, where oxygen diffuses from blood vessels into three-dimensional tissue. In the next chapter, we look at ways to numerically evaluate diffusion-type equations.

1.2.4 The Finite Difference Methods

While the diffusion equation derived above can be solved analytically, my program uses a 3D version of the equation with sources and sinks. The PDE setup in my code is very difficult to solve analytically. Therefore, we need to implement some kind of numerical evaluation method. Additionally, since the model we are building upon uses matrices/grids to represent data and visualize findings, it would be efficient to somehow utilize a similar grid in our numerical solutions.

Finite difference methods are a way to numerically evaluate partial differential equations using a discretized grid. In the simple case of one-dimensional space, we can use a 2-dimensional mesh grid to plot the function $U(x,t)$, where the axes represent the time and space dimensions. The general idea is to use initial and boundary conditions to fill in the function values across the grid one row at a time. Simply put, the finite difference methods use known function values at the previous time step to fill in the function values for the current time step. There are several finite difference approaches, such as the explicit and the implicit methods. In this project, I will use an explicit method described below. For future projects, it would be interesting to

implement the implicit Crank-Nicholson method that uses more linear algebra, which could potentially optimize the process.

1.2.5 Explicit Finite Difference Method

In this section, we go over the numerical evaluation method for 1-dimensional diffusion

$$U_t(x,t) = U_{xx}(x,t).$$

This part closely follows Lesson 38 of [10], so please see the textbook for more details.

1D Numerical Formula Derivation

We begin this method by constructing a 2-dimensional grid, where $x_i = ih$ and $t_i = ik$ for real numbers h, k. Figure 1.2.2 shows such a grid that was used in textbook ([10]). You can see the time and space dimensions represent the two axes on the grid, with h, k representing the grid sizes in x and t directions. In this example, we assume that each time step is weighted the same.

First, notice that the formula for the time-derivative of the temperature function gives us the approximate equation

$$U_t(x,t) \approx \frac{U(x,t+k) - U(x,t)}{k} = \frac{U_{i+1,j} - U_{i,j}}{k},$$

where $U_{i,j} = U(jh, ik)$. Note that k is a grid dimension, which, ideally, is small enough to allow this approximation. Now, let us use the grid to find a formula for the second space derivative of U.

Using the weighted average formula, we can write the derivative at the location x in terms of the previous and following locations on the grid. Namely,

$$\begin{aligned} U_x(x,t) &\approx \frac{U(x,t) - U(x-h,t)}{h}, \\ U_x(x+h,t) &\approx \frac{U(x+h,t) - U(x,t)}{h}. \end{aligned}$$

Then, we can approximate the second spatial derivative of U by using the two approximations above:

$$U_{xx}(x,t) \approx \frac{U_x(x+h,t) - U_x(x,t)}{h} = \frac{U(x+h,t) - U(x,t)}{h^2} - \frac{U(x,t) - U(x-h,t)}{h^2}$$
$$= \frac{U(x+h,t) - 2U(x,t) + U(x-h,t)}{h^2} = \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h^2}.$$

We can use these approximations in the heat equation by replacing $U(x,t)$ with $U_{i,j}$, where $t = ik$ and $x = jh$. Keep in mind that our goal is to solve for the value of U at the next time step. By making the substitutions for U_t and U_{xx} in the heat equation, we can solve for $U_{i+1,j}$ as seen below:

$$U_t = U_{xx}$$

$$\frac{U_{i+1,j} - U_{i,j}}{k} = \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h^2}$$

$$U_{i+1,j} = U_{i,j} + \frac{k}{h^2}(U_{i,j+1} - 2U_{i,j} + U_{i,j-1})$$

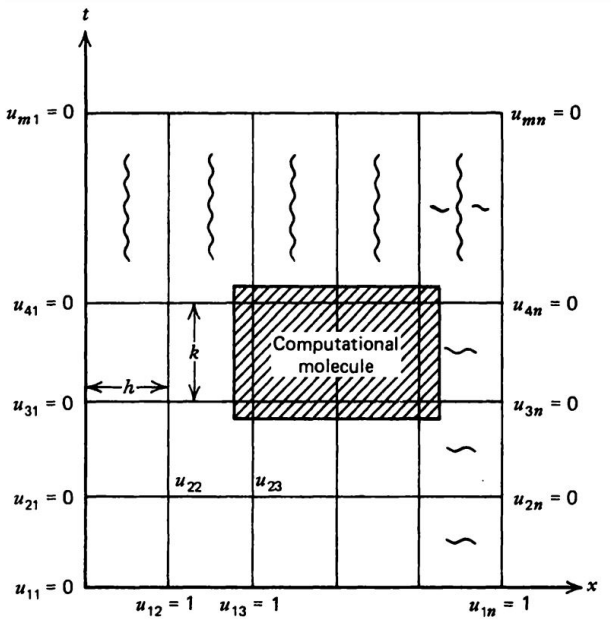

Figure 1.2.2: 2-D Grid for numerical approximation

The formula above is what we use to calculate the value quantity of heat at location $x = ik$ at time $t + 1 = (j + 1)h$.

2-D and 3-D Explicit Formulas

Deriving the explicit finite difference method formula for higher dimensions (2D, 3D) follows a very similar logic we used above to derive the 1D formula. Therefore, here we simply state the results for higher dimensions. In this project, I will be using a cubic grid, therefore we assume that the step size in all three dimensions is the same ($Dx = Dy = Dz$).

In 2 dimensions, we have the XY grid and the time dimension. Here, let $U_{i,j,t}$ represent $U(iDx, jDy)$ at time step t. Let us denote the step size in time by Dt for consistency purposes. Recall that in the 1D version, the step size in time dimension was denoted by k, but now we refer to it as Dt for consistency. Then, the oxygen concentration at point i, j at the next time step is given by

$$U_{i,j,t+1} = U_{i,j,t} + Dt\left(\frac{U_{i+1,j,t} + U_{i-1,j,t} - 2U_{i,j,t}}{Dx^2} + \frac{U_{i,j+1,t} + U_{i,j-1,t} - 2U_{i,j,t}}{Dy^2}\right)$$

as seen in section 2.3.4 of [23]. (Note that in our case, $\alpha = 1, \Delta x = \Delta y = Dx, \Delta t = Dt$.) Since $Dx = Dy$ we get

$$U_{i,j,t+1} = U_{i,j,t} + Dt\left(\frac{U_{i+1,j,t} + U_{i-1,j,t} + U_{i,j+1,t} + U_{i,j-1,t} - 4U_{i,j,t}}{Dx^2}\right) \tag{1.2.1}$$

For 3D grids, we modify equation 1.2.1 to include a third space dimension, z, represented by the index k. As shown in section 3.1 of [8], we can find oxygen concentration at the next timestep $t + 1$ at index i, j, k with the formula

$$\begin{aligned} U_{i,j,k,t+1} &= U_{i,j,k,t} \\ &+ Dt\frac{U_{i+1,j,k,t} + U_{i-1,j,k,t} + U_{i,j+1,k,t} + U_{i,j-1,k,t} + U_{i,j,k+1,t} + U_{i,j,k-1,t} - 6U_{i,j,k,t}}{Dx^2}. \end{aligned} \tag{1.2.2}$$

(Again, note that in a cubic grid $Dx = Dy = Dz$, and that Dt denotes the time increments.) Now that we have covered the important partial differential equations and numerical approximation formulas needed for our model, we conclude the math background overview.

1.3 Modeling Background

In this section, I explain different types of computational models, go over several oxygen diffusion models, and discuss the previous work of Dr. Norton's computational biology lab at Bard College.

1.3.1 Agent-Based and Continuous Models

Models and simulations are often used to observe biological processes, such as epidemics, population dynamics, immunotherapies, and more. These processes, in real life, can take years and decades to occur. Simulations, on the other hand, can be done in days, hours, and even minutes. They can also be less costly, and require fewer resources than conducting real-life research with live subjects. While models and simulations can be more efficient, they are also, by nature, imperfect. A model can generate results very close to the actual phenomenon, but most models require assumptions and simplifications that do not always follow in reality. For the purposes of this project, I look into two major types of models: agent-based and continuous. Specifically, I focus on agent-based and continuous models of cancer.

An agent-based model incorporates autonomous agents capable of making decisions based on the information they gather from their local environment without having access to the entire system. For example, an agent-based predator-prey model, where each animal is an agent. Prey agents would be checking their local environment to see if they can detect a predator in the vicinity, and move away from it, if possible. Predator agents, on the other hand, would be moving around randomly until they can sense a prey agent nearby, and move toward them. These decisions are made by agents based on what information they can gather from their immediate environment. Predator and prey agents do not have any information about the entire system. They do not know how many predator/prey agents there are, their locations, or the locations of food sources. This is close to how real-life agents make decisions. A predator or a prey does not have access to its entire environment and therefore has to make decisions based on what information is available locally. ABMs simulate the same decision-making we see in nature.

ABMs are used to simulate cancer as well. Usually, each cancer cell is programmed to be an independent agent with proliferation, migration, and death rates. This allows a researcher to track the development, growth, branching, and morphology of tumors on a cellular level.

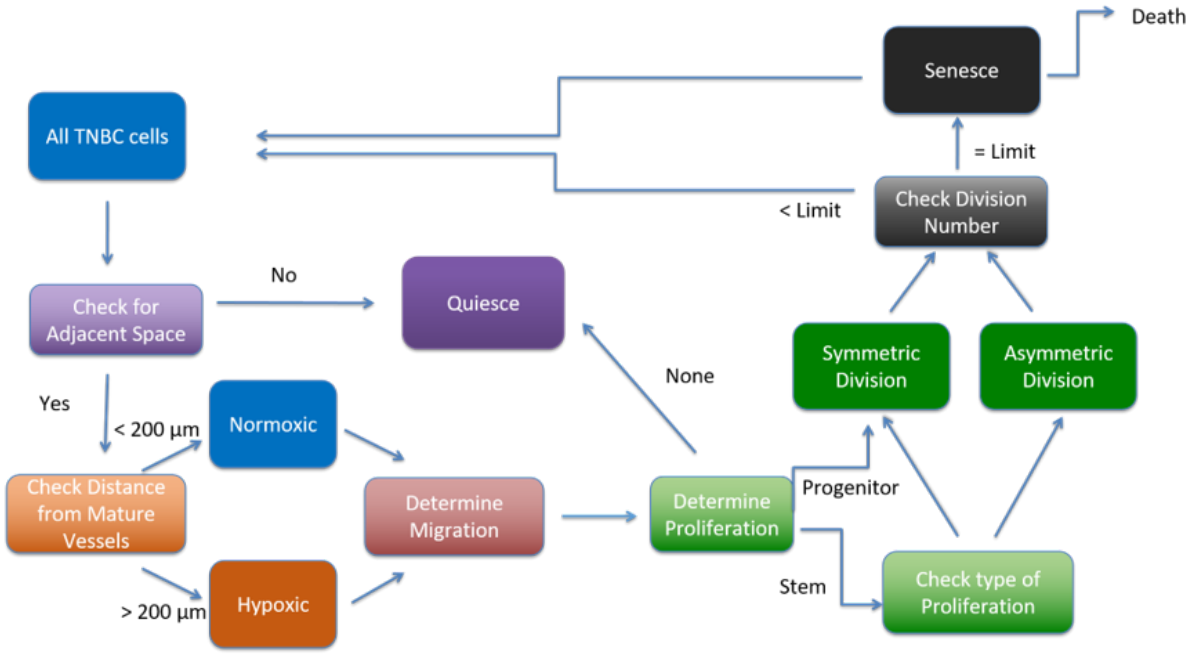

Figure 1.3.1: Sample ABM Flowchart

Each cell makes decisions based on some type of decision tree, where each decision depends on information gathered from the cell's parameters, or the environment. For example, Figure 1.3.1 from another cancer ABM ([40]) shows how each cancer cell agent makes decisions. At first, a cancer cell checks for empty space adjacent to it in a discretized grid. If there is no space (for example, if it is surrounded by other cells), the cell will be dormant. If there is space, the cell determines whether it is hypoxic or not, because being hypoxic affects migration and proliferation rates of cancer in this model. After determining the migration and proliferation rates, the cell will migrate accordingly, and be dormant if it can not proliferate. If it can, then a cell will divide symmetrically, if it is a progenitor cell. If it is a stem cell, then it will either divide symmetrically or asymmetrically depending on probability. Afterward, a cell checks if its division limit is reached. If it is, then it senesces and potentially dies off. Otherwise, the cycle repeats.

Agent-based models of cancer have led to interesting results and breakthroughs. To name a few, Gong et al. ([17]) found that tumor growth is insensitive to the initial location distribution of immune cells. A model by Norton and Popel ([40]) identified cell seeding rates and locations as important contributing factors to tumor growth and were able to give insight into conditions that lead to exponential and decremented tumor growth. Casarin and Dondossola ([4]) studied prostate cancer's response to R223 and found that it only got eradicated in medium or micro-sized tumors with ineffective results in large tumors. Heidary et al. ([21]) used an agent-based modeling approach to simulate the interactions between cancer cells and fibroblasts and found that the signaling of cancer cells caused the status of fibroblasts to switch in the tumor microenvironment. Olsen and Siegelmann ([42]) programmed an ABM of tumor angiogenesis, but unlike many other models, they included both cancerous and non-cancerous tissue. Their ABM showed that including non-cancerous tissue does not yield significantly different results from models that do not include such tissue. Novak et al. ([41]) used agent-based modeling to observe how provider-patient interactions influence the de-implementation of breast cancer screenings. Their ABM suggested that the rates of screenings are lower in women outside the range of 50-74 even when the provider recommendations remain unchanged. A study by Rivera et al. ([48]) used an ABM to predict the critical role of RAC1 (a specific gene) for the metastasis of ovarian cancer.

While our project intends to build upon an ABM, the module I added is continuous. Continuous models are models applied to continuous data and processes. Such processes occur at every point in time, as opposed to discrete intervals. For example, the military uses continuous modeling to simulate the trajectory of missiles. Biologists model population dynamics. Engineers use continuous models to simulate water levels when constructing dams. This type of model normally uses partial or ordinary differential equations. A very common math model that uses differential equations is the SEIR model in mathematical epidemiology. The SEIR (Susceptible-Exposed-Infected-Recovered) model simulated the transmission of a disease through a system of differential equations. Wickramaarachchi and Perera ([58]) used a typical SEIR model to de-

scribe Covid-19 dynamics in Sri Lanka. Using the total population N, exposure rate β, infection rate σ, recovery rate γ, and COVID death rate μ, they constructed the following system:

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta}{N}SI \\ \frac{dE}{dt} = \frac{\beta}{N}SI - \sigma E \\ \frac{dI}{dt} = \sigma E - \gamma I - \mu I \\ \frac{dR}{dt} = \gamma I \end{cases} . \tag{1.3.1}$$

Establishing initial conditions and evaluating this system allowed them to estimate optimal transmission rates (β), as well as initial parameters of the rate.

Much like ABMs, continuous models are a useful tool for computational cancer research. Dingli et al. ([7]) used mathematical modeling to check how varying parameters of tumor-virus interactions affected the therapy. Kozlowska et al. ([24]) used a combination of machine learning and math modeling to predict a patient's response to cancer immunotherapy. Gevertz et al. ([15]) used the model to explore how different types of drug resistance influence the spatial development of tumors. In our model, I use partial differential equations to simulate the diffusion of oxygen from the blood into cancerous tissue.

1.3.2 Modeling Oxygen Diffusion in Tumor

Real-life processes that include the diffusion of a substance are often modeled using partial differential equations. Therefore, oxygen diffusion in tumors is usually done with continuous modeling. Here, I do a brief overview of existing studies that used an oxygen diffusion model in cancer. A paper by Gevertz et al. ([15]), as mentioned above, largely inspired this project. They used two diffusion models: one to simulate the diffusion of oxygen, and one to simulate the diffusion of a drug. Their main goal was to observe how two different types of drug resistance affect the spatial dynamics and growth of cancer. I will focus on their oxygen diffusion module. Their partial differential equation has a form very similar to what I use in my project:

$$\text{Oxygen}_t = \text{Diffusion} \times \text{Oxygen}_{xx} + \text{Supply} - \text{Uptake}.$$

The supply term refers to the quantity of oxygen supplied by blood vessels at each spot. This is measured simply by checking how many blood vessels are within some radius of each

point in space. The uptake term refers to the quantity of oxygen absorbed by nearby cells. The diffusion is simply a second spatial derivative scaled by a dimensionless coefficient. They used sink-like boundary conditions. Their model had two space dimensions, so an important aspect of this project is to use the foundations of Gevertz et al. ([15]) to build a 3-dimensional model. Their study used a forward finite difference method to numerically estimate the values of their 2D oxygen concentration function. This project uses the same numerical method but instead evaluates 3-dimensional functions.

Many projects have focused on various other aspects of oxygen diffusion. For example, Pratx et al. ([45]) used a computational model to observe oxygen depletion, and how it affects the oxygen enhancement ratio. The model's predictions indicated that any changes in the capillary oxygen tension decrease the effects of FLASH (ultra high-dose rate radiotherapy). Li et al. ([26]) used a 1-dimensional diffusion model to simulate how oxygen diffuses in a biofilm. For their boundary conditions, they set the x-derivative of oxygen concentration to be 0 at the boundary, because oxygen is often depleted before it reaches the edge. While this study was observing the growth and thickness of the biofilm instead of focusing on cancer specifically, it is still a good example of how even 1-dimensional oxygen diffusion is relevant and necessary for modeling. Franko and Sutherland ([14]) used modeling to investigate the effects that limiting the diffusion of metabolites had on the development of necrosis in a tumor environment. Their model suggested that the number of hypoxic cells in spheroids grown at more than 5% oxygen was very small. Riffle and Hegde ([47]) used an oxygen diffusion model to observe how cancer cells adapted hypoxia in tumor spheroids. Liapis et al. ([27]) modeled oxygen diffusion in absorbing tissue, which is very similar to this project. They, however, used orthogonal collocation, a different numerical evaluation method, and introduced moving boundaries. Schultz and King ([50]), used a similar mathematical model of oxygen diffusion and reaction with flux boundary conditions, where the consumption of oxygen by cells in the biological system was determined by Michaelis-Menten. Tannock ([55]) studied the relationship between oxygen diffusion and cellular radiosensitivity in tumors by analyzing cell survival curves for different types of tumors. Simpson

and Ellery ([52]) derived a new analytic solution for a steady-state model of oxygen diffusion with nonlinear uptake of spherical cells. In conclusion, various papers have used continuous modeling and PDEs to study oxygen diffusion in tumors.

1.3.3 Previous Work

In this section, I discuss previous work done in Dr. Norton's lab that lead to the development of this project. Note that this section will not focus exclusively on angiogenesis modeling. Instead, I will overview some of the previous related studies that used computational models to study triple-negative breast cancer (TNBC).

A previous paper by Norton et al. ([38]) used a similar computational model to observe the interplay between CCR5 cells, cancer stem cells, and hypoxia. Stem cells are types of cancer stem cells that have unlimited replicative ability. In other words, they can produce uncontrollably and have been known to cause tumor relapses. The simulation started with all cells checking for adjacent empty spaces; then, the migration rate was determined (CCR5+ cells had 10 times higher migration rates compared to other cells). Then, the model determined proliferation with 50% of progenitors and 20% of stem cells proliferating a day; the division of stem cells was based on random probability (5% for symmetric and 95% for asymmetric). After proliferating, the cells either repeated the cycle or died based on whether they had reached the proliferation limit. This study found that anti-stem cell treatment decreased the tumor size, but only temporarily. Even one stem cell was enough to eventually regenerate the tumor; thus, the total elimination of stem cells is necessary for the lasting effects of the treatment. They also found that stem cell proliferation rates have a more significant effect on tumor growth than migration rates.

Another paper by Norton et al. ([37]) used a computational model to study the effects of stromal macrophages, fibroblasts, and tumor vasculature on TNBC heterogeneity. Their models included TNBC, angiogenesis, and stroma modules. In each iteration of the TNBC module, the cancer cells check for empty space and distance to a mature vessel. If a mature vessel is detected, they determine whether the cancer cell is hypoxic based on its distance from the

vasculature. Cells' migration was determined using hypoxia status. Then, the model determined the proliferation of tumor cells, and depending on whether the cell is a stem or progenitor, it divided either symmetrically or asymmetrically. Finally, the division number of a cell is checked (which is reduced every time the cell proliferates). If the number is less than the cell's limit, the cycle continues; otherwise, the cell is subject to senescence and possible death. The initial model included a $1000 \times 1000 \times 1000$ grid with 100 cancer cells and 8 blood vessels. Macrophages and fibroblasts were then randomly inserted into the grid. This model was used to conclude that increasing stromal effects on proliferation rates of TNBC cells actually decreases tumor growth. They also found that a small number of macrophages increased the size of the tumor, while larger (~ 1000) numbers inhibited the growth due to space restrictions.

A third paper ([56]) based on a computational model of TNBC by Norton in 2020 added a new T-cell immunotherapy module to the ABM. The goal of the research was to observe how the tumor responded to the T-Cell treatment. Their model included tumor (progenitor and stem cells), macrophages, expanding vasculature, and immune cells. Their research found that while the T-cell treatment successfully decreased the size of the tumor, the cancer was still not fully eliminated. Therefore, they observed recurrence of cancer proliferation in the aftermath. My colleagues and I later expanded on this module by simulated a different type of immunotherapy of TNBC.

In that research I conducted with colleagues in Norton's computational lab, we focused on adding a new tumor immunotherapy known as the Chimeric Antigen Receptor (CAR) T-cell therapy. Our research aimed to test the effectiveness of immunotherapy under varying expressions of antigens within tumors. CAA, or, the cancer-associated antigen is an antigen bound to tumor cells and is used to target the tumor. We programmed two distinct models: one that used a binary expression of antigens where each tumor cell had 0% or 100% of presenting the antigen ([11]) and one that used a gradated expression of antigens, where the probabilities of tumor cells expressing the antigen were distributed on a bell curve ([16]). Our research showed that CAR T-cell therapy successfully reduced the size of the tumor in both binary and gradated

models, with a higher success rate for complete elimination of the tumor in the gradated model compared to binary. The studies mentioned above led to the development of my senior project, which aims to improve the simulation of hypoxia in the existing model.

2
Computational Methods

Dr. Norton's computational lab has developed previous agent-based models of triple-negative breast cancer ([11], [16]). Those previous simulations include tumor cells, vasculature (blood vessels), immune cells, data collection, and other cells in tumor microenvironments. Those models also keep track of hypoxia levels over time, where a cancer cell is considered hypoxic if it is further than 200 microns away from vasculature. That is a very simplified way of measuring hypoxia. It is important to have a more realistic way of checking whether a cancer cell is hypoxic, as hypoxic cells have lower migration rates, which could affect the outcome of simulations. The goal of this project is to incorporate an oxygen diffusion module into the existing ABM. I add code to use vasculature as an oxygen source, and diffuse oxygen into the model from each vasculature node. Now, we have access to the oxygen concentrations at the location of every cancer cell, meaning that we can measure hypoxia levels more realistically.

The model I expanded in this project uses a $500 \times 500 \times 500$ matrix to keep track of the locations of the vasculature where each grid unit represents 2 μm. Though the final grid is $500 \times 500 \times 500$, I used a reduced version of size $50 \times 50 \times 50$ for testing and better visualization purposes. The first step was to write 2D diffusion code, and eventually increase it to three dimensions. Simulation of vascular diffusion occurred in 4 steps: 1. 2D point source diffusion, 2. 2D line source diffusion, 3. 3D cubic patch diffusion, and 4. Vasculature diffusion. I first

took a 2D grid of the size 50×50 and put a single point source in the middle to observe the diffusion. Then, I tested line diffusion on the same grid by taking the entire top row of the grid as a source. Both of these 2D models were validated using existing programs and resources to ensure my results were comparable to those existing resources. Then, I moved to 3D diffusion in a $50 \times 50 \times 50$ grid by placing a cubic patch in the center, after which I validated my results using existing diffusion models. Finally, once my 3D program was validated, I used the tumor model developed in Dr. Norton's computational biology lab to obtain locations of vasculature and used those as sources. In other words, I inputted initial vasculature as oxygen sources and had them diffuse into the model.

2.1 Code Overview

This section will cover the setup of the code. I programmed the entire project in ©Matlab R2020b ([33]). I first set up our 50×50 Oxygen matrix U and set $Dx = Dy = 1$. Then, I place a heat source of 100 non-dimensional units at the center with coordinates $(25, 25)$. Next, I introduce our boundary conditions: I fixed the boundaries at 0 to cut off the oxygen. The tumor model I plan on building upon simulates a section of a solid tumor, so I decided to fix the bounds. An important next step is deciding what the time step should be. In the mathematics background section, I discussed the classic heat equation of the form $U_t = U_{xx}$. In practice, the equation actually has the form $U_t = \alpha U_{xx}$, where α is a heat diffusion coefficient that has a significant effect on the specifics of the diffusion. It can determine how fast and how much diffusion takes place. The numerical solution changes only slightly: Equation 1.2.1 becomes

$$U_{i,j,t+1} = U_{i,j,t} + \alpha \times Dt \times \frac{U_{i+1,j,t} + U_{i-1,j,t} + U_{i,j+1,t} + U_{i,j-1,t} - 4U_{i,j,t}}{Dx^2}, \tag{2.1.1}$$

where Dx is the grid width, and Dt is the time step. See Chapter 2 of [25] for more details.

I can decide what time step Dt and what α I want to pick. The smaller the time step, the more accurate the results will be. However, the smaller the time step, the slower the simulation. Langtangen and Linge ([25]) discussed the relationship between Dx, Dt, and α in their literature.

In Section 3.4, they define the Numerical Fourier Number as

$$F = \frac{\alpha Dt}{Dx^2}.$$

They state that in order for the solution to be stable, a sufficient condition for stability is $F \leq 0.5$. This can be rewritten as a condition of Dt, where

$$Dt \leq \frac{Dx^2}{2\alpha}.$$

Since $\alpha = 3 > 1$ in my program, I used $Dt = \frac{Dx^2}{2\alpha^2}$. This concludes the initial setup. The 3D code was vectorized for efficiency purposes, meaning that the calculations were performed on entire rows of entries at once, instead of looping through every individual entry. 2D code was not vectorized, however, since the program was fast enough unvectorized and slightly easier for a user to follow.

Inside a double loop, I went through every i, j index of our matrix U except for the boundaries. I updated the old value at position U_{ij} according to the Equation 2.1.1 restated below:

$$U_{i,j} = Ui, j + \frac{\alpha Dt}{Dx^2}\left(U_{i+1,j} - 2U_{i,j} + U_{i-1,j} + U_{i,j+1} - 2U_{i,j} + U_{i,j-1}\right).$$

Once I updated the value, I calculated the absolute change by looking at the difference between the old and new values of $U_{i,j}$. I kept track of the cumulative change. I repeated the process until the cumulative change is less than a small percentage of the highest starting value of U. That means I terminated the program after it reached a steady-state and further diffusion became negligible.

2.2 2D Diffusion of a Point

First, I placed a point source in the middle of a 2-dimensional 50×50 grid and make it an oxygen point source with a supply rate of 100 non-dimensional units. I set the boundary conditions to be 0 on all four boundaries of our grid. I then diffuse it until it reaches a steady state. I considered the program to have reached a steady state when the maximum change in oxygen concentrations across the grid from one time step to the next was less than 0.01% of the starting

highest value of all oxygen concentrations (in our case 100). I expect the point to diffuse in the shape of a circle centered at $(25, 25)$. I also expect the concentrations to get lower and lower the further out into the grid we go. I also expect the edges to stay at 0 based on our boundary conditions. After the program terminates, I validated my model.

2.3 2D Diffusion of a Line

After I finished and validate the point source (0-dimensional source) diffusion, I wanted to make sure our program works in the case of a 1-dimensional source (a line) as well. Thus, I took the same 50×50 grid but put a line as a source instead of a point. I chose the top edge of the grid to be a line source. In other words, every point on the line $y = 1$ has a starting value of 100 instead of 0. After setting the source line, I specified the rest of the boundaries to be 0, as before. I then let the line diffuse downward until it reached a steady state. This was an important step, the goal for this project is to successfully diffuse oxygen from vasculature, which, when simplified, could look like line segments in either 2D or 3D. Therefore, testing and confirming the correctness of line diffusion in 2D means that we can now simulate vasculature in 2D using line segments.

2.4 3D Diffusion of a Cubic Patch

After completing the diffusion of both 1-D and 2-D sources in a two-dimensional grid, I moved up to three spatial dimensions. Although the grid I ultimately used to insert vasculature into will be the size $500 \times 500 \times 500$, for model validation and visualization purposes I reduced it to $50 \times 50 \times 50$. I centered a $7 \times 7 \times 7$ cube in the center of the 3D grid. This is an arbitrary size, although a smaller size would have been too small for a grid this large. I set every point inside that cube to be a source with the value of 1000. I increased the supply rate from 100 to 1000, since the grid is now 3D, and I wanted the diffusion to be easier to visualize. Note that just like with 2D diffusion, I needed to slightly modify Equation 1.2.2 to include the diffusion coefficient, obtaining

$$U_{i,j,k,t+1} = U_{i,j,k,t} + Dt \times \alpha \frac{U_{i+1,j,k,t} + U_{i-1,j,k,t} + U_{i,j+1,k,t} + U_{i,j-1,k,t} + U_{i,j,k+1,t} + U_{i,j,k-1,t} - 6U_{i,j,k,t}}{Dx^2}. \quad (2.4.1)$$

I ran the program until the diffusion reached a steady state, meaning that the maximum change in the oxygen concentration matrix from one timestep to the next was no more than a small percentage (0.01%) of the highest entry in the starting grid (1000). After the program ended, I validated my results using available tools and resources.

2.5 Obtaining the Diffusion Radius

As I discussed earlier, the diffusion coefficient and the diffusion time step have a big impact on the characteristics, speed, and spread of the diffusion. Since this is a computational biology project, I have some knowledge of what the correct diffusion radius is supposed to be. In other words, as seen in [9], the normal diffusion radius for cancer cells is between 100 and 200 microns. That means if a cancer cell is not within that range from an oxygen source, it will be considered deprived of oxygen, and therefore hypoxic. I measured the diffusion radius by looking at every non-zero oxygen concentration location in the matrix U, then measured distances from all those locations to the central point source, and returned the maximum of those distances. In other words, I measured the maximum distance from the diffusion source to all non-zero concentrations. In our simulation, after trial and error, I set the diffusion coefficient α to 4 and the time step to $\frac{Dx^2}{2\alpha^2} = 0.03125$ to obtain a diffusion radius of 180 microns from a point source centered inside the $500 \times 500 \times 500$ grid. I then kept the same parameters as I used the model in the future. However, note that our time step is rather small, which means that I exchanged the speed of the program for accuracy.

2.6 3D Diffusion of Initial Vasculature

Finally, I inputted blood vessels into our program as initial sources and have them diffuse throughout the grid. This is the last step in our diffusion trials because once I successfully

diffused oxygen from the vasculature into the grid, I added tumor cells into the system. I used a model, developed in Dr. Norton's computational biology lab, that creates 8 initial capillaries inside the grid. Then, the code fills a $500 \times 500 \times 500$ grid with logical values (logical values are 0s or 1s): 0, if there is no vasculature in that location and 1 if there is vasculature in that spot. I used the code to create capillaries and the code to fill in the vasculature grid. I then used that grid to find which locations in our program need to be made a source with starting value of 1000. After I put sources into every location with vasculature in it, I ran the diffusion. As of now, I have not made the vasculature a continuous source - I am simply doing a one-time diffusion to test the program. Another thing to note is that we consider starting blood vessels to have already undergone anastomosis (see Angiogenesis Section of this paper). Therefore, we can assume that each of the vessels in the vasculature diffuses oxygen. I ran the program where oxygen diffuses from the vasculature with no sinks or sources.

However, I want to eventually make vasculature a continuous source. Our blood vessels continuously supply oxygen, therefore, I need to add a supply function to our code. So, I ran the program a second time with two added functions: supply and uptake. Let $\text{Supply}_{i,j,k,t}$ represent the supply at location i, j, k at time t. In other words, this function quantifies the amount of oxygen supplied at this location by vasculature at a given time. I calculated this by looking at the 8 neighbors of the location i, j, k. Then, I counted how many of these neighboring locations currently contain a blood vessel node. I then multiplied that number by 1000 (supply rate of vasculature). I then multiplied that number by the timestep Dt, and add it to the current amount of oxygen concentration at the location i, j, k, as shown on Page 7 of [25].

The last thing I needed to add is cellular uptake. Once oxygen is diffused into the tissue, cancer cells absorb oxygen and nutrients in order to migrate and proliferate. I implemented a cellular uptake function similar to the one used in [15], where at each location at a given time step, they count how many cancer cells are within a certain range. Then, I scaled up that number by a cellular uptake constant. Finally, the uptake is going to be the minimum of the current oxygen concentration at this location, and the scaled number of cells within the sensing

range from this location. This makes sense because if there are too many cells concentrated at the location, their oxygen uptake can not be more than what is currently available at this location. In other words, the uptake function Uptake can be expressed as

$$\text{Uptake}(i,j,k) = \sum_{\text{C in Cells}} \begin{cases} 1 & \text{if } ||C_{\text{location}} - (i,j,k)|| < R \\ 0 & \text{otherwise} \end{cases}, \tag{2.6.1}$$

where $R = 3$ is the sensing radius of cells, and Cells is the collection of tumor cells.

Adding supply and uptake functions makes our modified Finite Difference Method formula look slightly different from formula 1.2.2:

$$\begin{aligned} U_{i,j,k,t+1} =& U_{i,j,k,t} + \text{Supply}(i,j,k) \cdot Dt + \text{Uptake}(i,j,k) \cdot Dt \\ &+ Dt \cdot \alpha \left(\frac{U_{i+1,j,k,t} + U_{i-1,j,k,t} + U_{i,j+1,k,t} + U_{i,j-1,k,t} + U_{i,j,k+1,t} + U + i,j,k-1,t - 6U_{i,j,k,t}}{Dx^2} \right). \end{aligned}$$

Since the rest of the code is vectorized, meaning, the program operates on entire arrays at once instead of looping through individual entries, the code for supply and uptake also needs to be vectorized. Ideally, they will operate on an array of locations, and return an array of values. Before vectorizing the code, I tested the unvectorized versions. For the supply function, I placed a 5×5 rectangle (unfilled) in the center of a $50 \times 50 \times 50$ grid to represent vasculature. In other words, I temporarily edited the vasculature grid to only represent a rectangle, to see the heatmap of the supply function. Since we can calculate the intended values of supply by hand, it can be manually validated to be correct. We would expect to have no supply at grid locations more than one row/column away from the central rectangle. We would also expect to have the most supply around the corners of the initial rectangular vasculature.

2.7 Model Validation

Before I can use the model for any research purposes in the future, it is important to first validate the program by showing how the results compare to already established studies or programs. For the 2-D source, I used an online Python tutorial ([35]), that aims to obtain a numerical

solution to the 2D heat equation. Their code diffuses the heat from the top line ($y = 0$) of their grid downward. To compare it to our model, I set their grid size to be the same as ours and set our diffusion parameters (Dx, Dy, α, Dt) to match theirs. I then ran both programs and calculated the average error. Note that their Python program also uses the finite difference method as they loop through their heat concentration array. For the 3D model validation, I used the program found at [43]. Their vectorized code includes a steady-state mode using the steady-state Laplace's Equation. This resource was also used as a guide for vectorizing our code. I increased their grid size to $500 \times 500 \times 500$ and modified their initial conditions to also include the 3D cubic patch in the center. After both programs reached a steady state, I calculated the average percent error across the two matrices.

3
Results

This project aims to create an oxygen diffusion module with the purpose of adding it to existing agent-based breast cancer model(s). I programmed four diffusion models to lead up to the main goal, which is a vascular diffusion of oxygen. I started with a simple point source diffusion in 2D, then a line source diffusion in 2D, then a cubic source diffusion in 3D, and ended at 3D oxygen diffusion from the vasculature. In this section, I go over the visualizations and findings of each of the diffusion models mentioned above.

3.1 2D Diffusion of a point

Figure 3.1.1 shows the 50×50 grid after a steady-state diffusion of a point in the middle of a 2D grid. I see that as expected, the highest concentration is in the middle, and lessens as we move outward. I also see that the boundaries are set at 0, as I programmed. Since the diffusion is not biased in any direction, it makes sense for the point source to diffuse equally in every direction. That's why seeing the diffusion in the shape of a circle makes sense, as it is perfectly symmetrical in every direction. In other words, any set of equidistant locations from the central source has the same concentration of oxygen. The concentric circles in the figure demonstrate that very concept. Figure 3.1.2 shows a heat plot of the 2D heat diffusion matrix obtained from [35]. Recall that the initial and boundary conditions were matched to our program, as well as the size of the grid. The two figures look very similar visually. One thing

to note is that the Python code I compared to my program was not a steady-state program. As in, the creators of that code were not checking to see if the change in a matrix was lower than a certain percentage. Therefore, to match their setup, I temporarily commented out the check for a steady-state condition I had in my program. We see a very similar diffusion in this figure, with expanding concentric circles around the central point source with decreasing values of oxygen concentration. I then calculated the average percent error between my model and the gold standard in order to confirm the correctness of my program. After calculating the average percent error between the two matrices, I got 0.3932%. This concludes the point source validation.

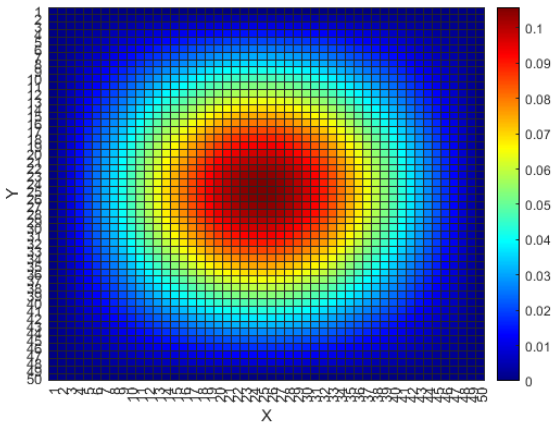

Figure 3.1.1: TG Model of 2D Point Diffusion

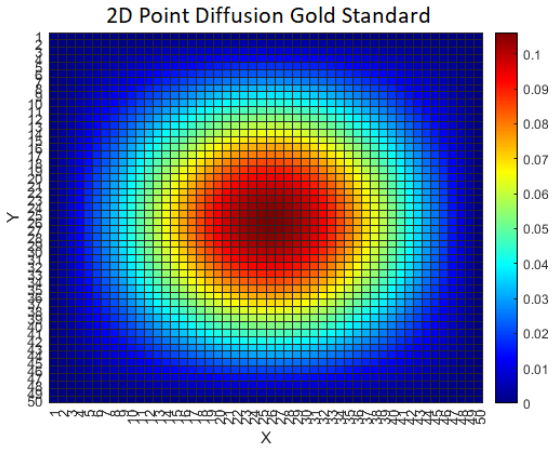

Figure 3.1.2: 2D Point Diffusion Gold Standard From [35]

3.2 2D Diffusion of a Line

Figures 3.2.1 and 3.2.2 shows the steady-state heat plots of our model and the gold standard from [35] after diffusion of a line at the top of a 50×50 grid. Starting values of every source point on that line were set to 100 in both models. Visually, our model looks as expected: I start at high concentrations at the top border and gradually decrease in the shape of a semicircle as I progress downward. The borders are fixed at 0, which aligns with our boundary conditions. It is important to note that since the supply line is at the top border of the grid, it can not diffuse

upward. That is why the diffusion visually we only see half of the diffusion. Otherwise, it would have been symmetrical in the upward direction as well.

This diffusion shape matches what I see in the gold standard version (Figure 3.2.2. We see on the heat bar that the values range from 0 to 100, starting with the highest concentrations at the top and diffusing downward. To check how similar my program was to the gold standard in [35], I calculated the average percentage error between the two final matrices after 750 iterations and got 1.3 % average error across 2500 entries. The Python program only did 750 iterations, so I modified my code to also run for 750 iterations only.

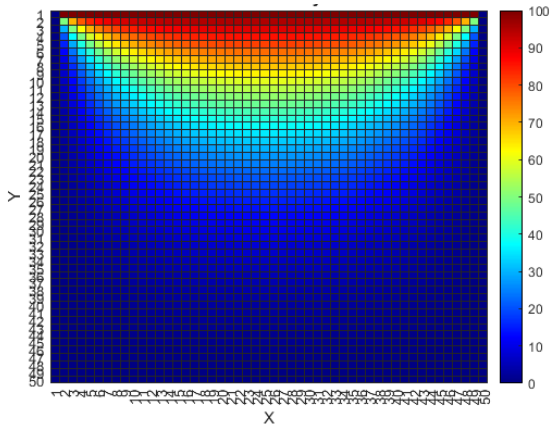

Figure 3.2.1: TG Model of 2D Line Diffusion

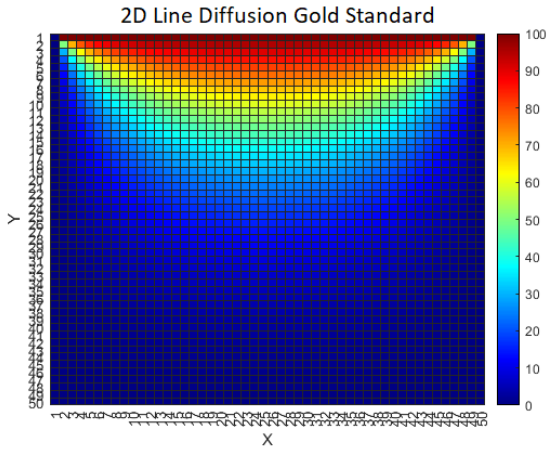

Figure 3.2.2: 2D Line Diffusion Gold Standard From [35]

After calculating average error, I was curious to see the distribution of individual errors, and whether the distribution would be uniform or follow any pattern. Figure 3.2.3 shows the distribution of errors. I see that the error between my model and the gold standard ranges between 0 % and 4%, with most of it concentrated at the bottom of the grid, just above the boundary. This can be explained by the fact that our diffusion in this case begins at the top border of the grid, so the error accumulates as it diffuses downward. In other words, our numerical evaluation method uses previous entries and rows in the matrix is calculate future rows and values. Therefore, the error builds up from one row to the next. That explains why most of the higher error values are concentrated in the very last row. We also see no error on

the boundaries, as they are fixed in both cases to be 0 and therefore are identical. Next, I move to the $500 \times 500 \times 500$ matrix in three dimensions.

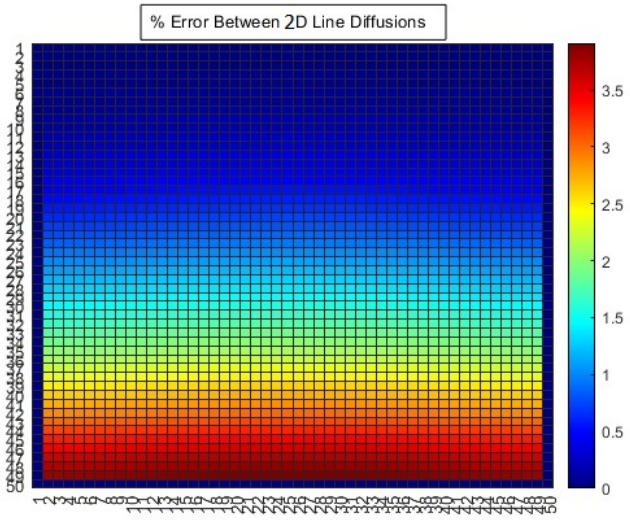

Figure 3.2.3: Heatmap of Percent Error In 2D Line Diffusion

3.3 Obtaining the Diffusion Radius

As mentioned in Methods, the normal range for oxygen diffusion from vasculature is between 100 and 200 microns ([9]). It is important for the accuracy of the program to make sure that our model can reach a realistic diffusion radius. If the oxygen does not diffuse enough in the program, then we might end up with incorrect hypoxia rates. Since our time step is very small, in order to save some runtime, I ran our program until I reach the lower bound of 100 microns. Note that our grid width represents 2 microns. Therefore, in terms of our grid width, our target diffusion radius is at least 50. Figure 3.3.1 shows such diffusion in 2D. This is actually a 2D slice from a 3D grid. The diffusion value was set to 4, the time step was set to 0.031, and the program terminated with the maximum change in oxygen concentration across the grid from one iteration to the next was about 0.0000007. As seen in the figure the diffusion begins in the center at position 250 and diffuses outward until somewhere around 300 and 325.

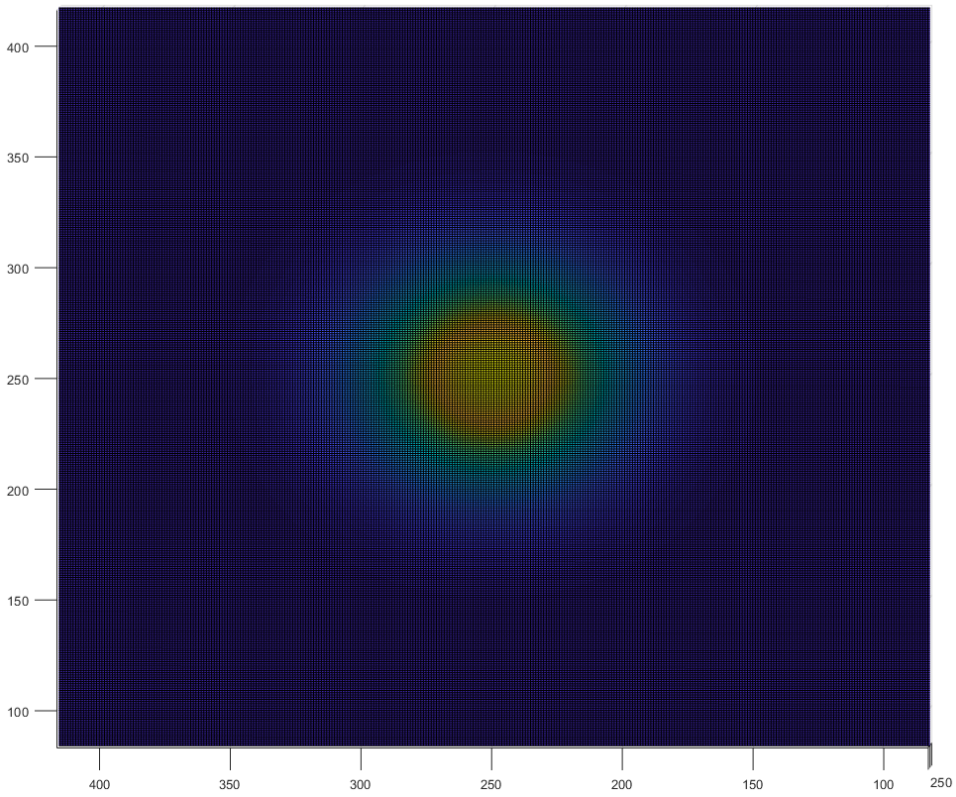

Figure 3.3.1: Diffusion Radius of 100 Microns:2D

You see the initial source is placed at locations (250,250,250), with the diffusion radius between 50-75 (100-150 microns). Now that the diffusion radius is within the normal range, I moved on to diffuse oxygen in a 3-dimensional grid.

3.4 3D Diffusion of a Cubic Patch

As mentioned in the Computational Results section, I set the grid size to $50 \times 50 \times 50$, and placed a cubic patch of the size $7 \times 7 \times 7$ at the center, where each point inside the cube has the initial value of 1000. Once again, a cube of side length 7 was arbitrary, but I found that 3 was too small for the grid. I then diffused it until the program reached a steady state, and compared the final oxygen concentration matrix to the one given by the 3D heat diffusion gold standard [43].

Figure 3.4.1 shows our program's steady-state distribution, which, as expected, is highly concentrated in the center and gradually decreases as we move toward the borders. Since our

data was 4-dimensional (three spatial coordinates, plus the fourth dimension for concentration values), the best way to visualize it was to take three central slices that demonstrate central diffusion. Note that our boundary conditions on the boundaries were still fixed at 0. As in the case of 2D diffusion, we have symmetry in 3D as well. I did not introduce any biased diffusion in any direction, meaning that the cubic source in the center diffused equally in every direction. So, if the source had been a single point, the diffusion would have been shaped as a perfect sphere. With the cubic source, we still see symmetry. The diffusion looks the same in every direction of the slice. My model visually matches the gold standard model shown in Figure 3.4.2. We see the same distribution of oxygen, concentrated in the center and equally damping out in every direction. As in the case of 2D diffusion models, I calculated the average error as a percentage and got 0.2456%. This percentage is small enough for me to conclude my 3D model validation. Next, I move on to input the initial capillaries into our program.

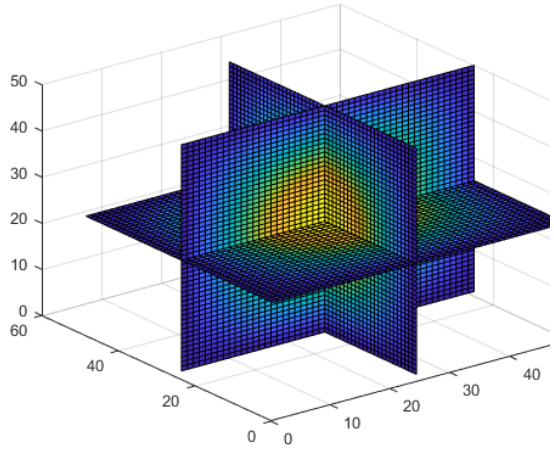

Figure 3.4.1: TG Model of 3D Cube Diffusion

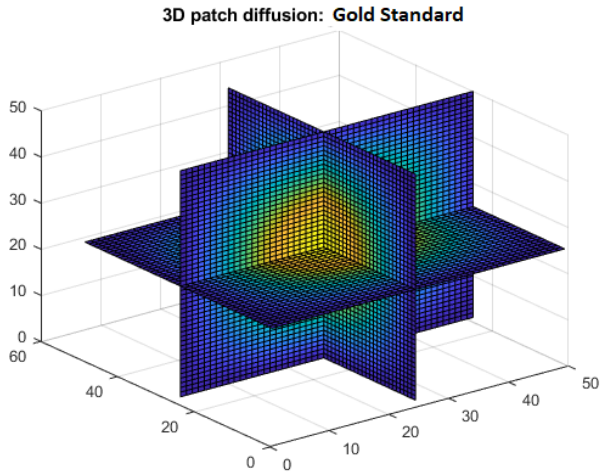

Figure 3.4.2: 3D Cube Diffusion Gold Standard From [35]

3.5 3D Diffusion of Initial Vasculature

The aim of this project is to be able to diffuse oxygen directly from vasculature instead of arbitrary point or line sources. So, after programming and checking the correctness of point source, line source, and cubic patch, I moved to vascular diffusion. I started by inputting the

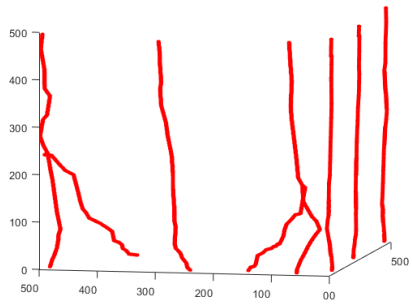

Figure 3.5.1: Initial Vasculature

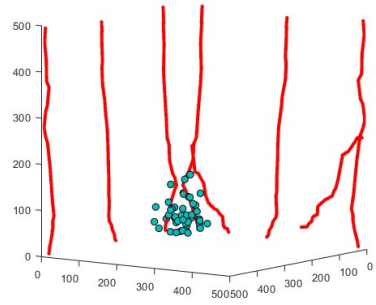

Figure 3.5.2: Setup of Tumor Cells

initial vasculature into the grid as initial sources. I then fixed the boundary conditions. The initial vasculature was taken from previous models developed in Dr. Norton's lab ([11], [16]).

Figure 3.5.1 depicts the blood vessels at the beginning of the program. We see that the initial blood vessels are spread near the boundaries of the grid. We assume that all these blood vessels diffuse oxygen. Recall that in general, sprouting blood vessels have blood flowing if they have undergone anastomosis. In this project, the initial blood vessels we are using as sources are assumed to have blood flow, thus, we use every blood vessel as oxygen source without having to check for anastomosis. I ran the program without sinks or sources. That means that there are no cancer cells to uptake the oxygen, and the blood vessels aren't programmed to continuously supply oxygen. Once the oxygen is fully diffused from the vasculature, I take slices of our oxygen concentration matrix through various vessels and locations to observe the distribution of concentrations.

Figures 3.5.3, 3.5.4, 3.5.5, and 3.5.6 represent these cross-sections of the oxygen concentration matrix diffused directly from the vasculature. The red lines represent close-ups of blood vessels shown in Figure 3.5.1. The first two figures, Figure 3.5.3 and Figure 3.5.4 were captured close to the edge of the grid. We can see that the oxygen diffusion is cut off at the edges and does not diffuse symmetrically around the vessels. We can still see gradients of oxygen concentration around the vessel, as expected. Once again, these figures do not include continuous supply and cancer cell uptake functions - these are steady-state one-time diffusion instances of oxygen into

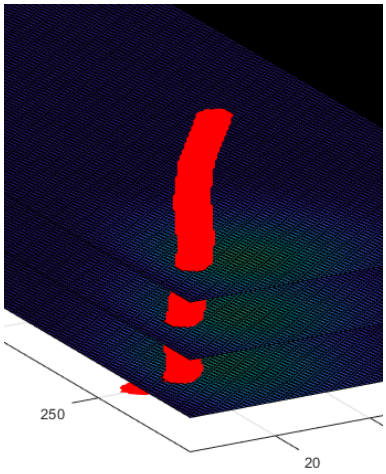

Figure 3.5.3: Diffusion From Vasculature

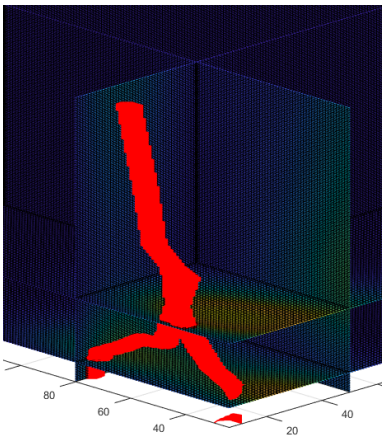

Figure 3.5.4: Diffusion From Vasculature

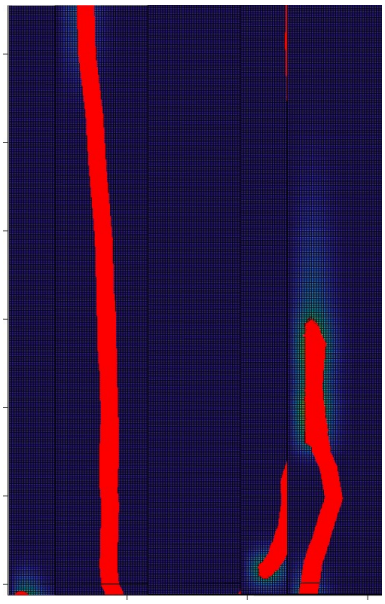

Figure 3.5.5: Diffusion From Vasculature

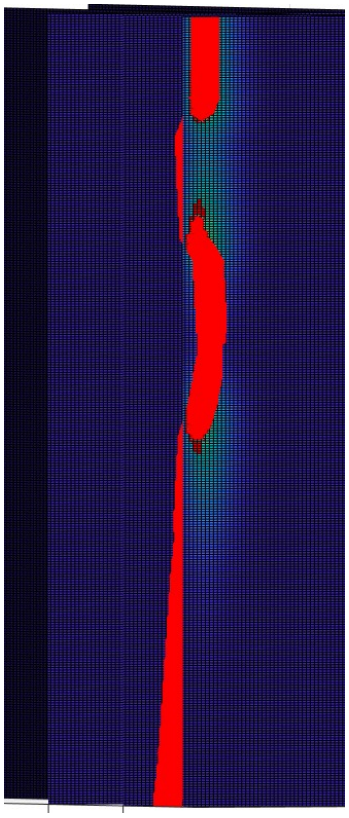

Figure 3.5.6: Diffusion From Vasculature

tissue. The diffusion radius is closer to the lower bound of the normal oxygen diffusion radius of the 100-200 micron range. The bottom two figures, Figure 3.5.5 and 3.5.6 show instances of diffusion where vasculature goes through a plane. Specifically, Figure 3.5.5 shows a piece of a vessel (in red) on the left side going through a cross-section of the grid. On that cross-section, we see a gradient of oxygen diffusion right around the intersection point between the vessel and the plane. In this case, the vessel is not close to the edge of the grid, so we can see a wider diffusion with a bigger radius. Figure 3.5.6 shows a case where a vessel goes in and out of a plane. We see that the entry and exit points have gradients of diffusion around them, as those locations are the closest to the oxygen supply. The area on the cross-section in between the entry and exit points of the red blood vessel seems to have lower oxygen concentrations due to the lack of color gradient, which makes sense as the vasculature is further away from that area.

Being able to diffuse oxygen from vasculature was one of the main goals of this project. Now that we have accomplished this, the next steps are to add cellular uptake and continuous supply options.

3.6 Adding Cancer Cells And Continuous Sources to the Model

As discussed earlier, another important goal of this project is to be able to have tumor cells interact with vasculature by absorbing the concentration of oxygen available at their locations. I was able to implement the code for the cellular uptake and vascular supply functions, however, they did not end up fully finished in time and thus did were not put in the final model. The draft of the code for both functions can be found in the Appendices. As mentioned in Methods, the supply function takes in a point and counts how many of its neighboring locations act as oxygen sources. Each cell has 27 neighbors in 3 dimensions (28, since we include the point as its own "neighbor"). So, a grid point that is surrounded by vasculature will have a higher amount of oxygen supplied at that location compared to a grid point far away from blood vessels. The uptake function counts how many cancer cells are within sensing distance from a given grid

location. If a point is surrounded by cancer cells, then the uptake value at that point will be high since many cancer cells are able to absorb oxygen from that location.

Since the 3D diffusion code (see Appendices) is vectorized, the supply and uptake functions need to be vectorized in order to be inserted into the program. Vectorizing those functions would be a potential next step for this project. I was able to test the current unvectorized version of the supply function. Figure 3.6.1 shows the placement of the initial vasculature in this test model. To simplify the location and shape of our initial vasculature, I modified the blood vessels, for now, to look like a rectangle in the center of the grid. It is easier to visualize than the complex shapes of blood vessels. We can see the yellow rectangle in the center. Figure 3.6.2 shows a heatmap of the supply function applied to every position on the grid. So, every point in that grid represents the supply value of the function at that location. The result looks as we would expect. The corners of the initial rectangle have a supply value of 3 non-dimensional units, one for themselves, and 2 for their 2 neighbors. The 2 central points on all four sides have supply values of 4, 1 for themselves and 3 for their three neighbors (diagonals count). We also see the supply is zero everywhere more than one row/column away from the initial vasculature, which is what we would expect. We also see a 2×2 square in the center with 0 supply, which also makes sense as that square is not directly adjacent to the rectangle. We can conclude from this figure that the 2D supply code is distributing the supply values as expected.

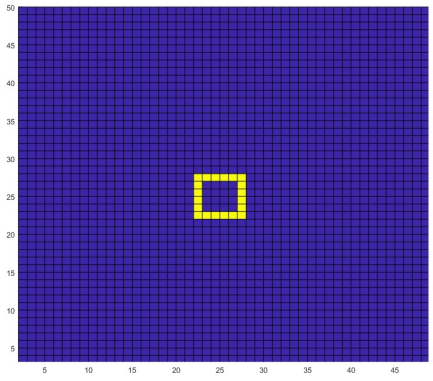

Figure 3.6.1: Supply Placement On Grid

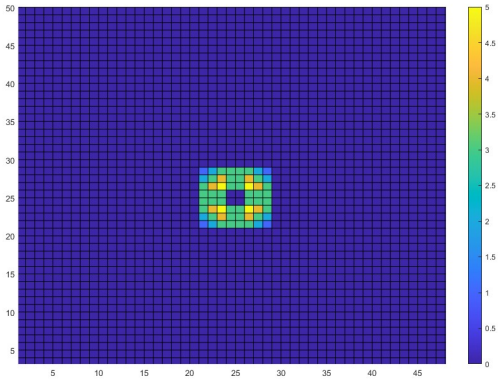

Figure 3.6.2: Heatplot of Supply Function

As I mentioned above, the supply and uptake functions were not implemented in the final program. Some of the issues I had with the code had to do with the difficulty of vectorizing those functions. Another issue I had was that the oxygen supply and uptake numbers generated by my functions were considerably larger than the available oxygen concentrations. That could have been due to incorrectly selecting supply and uptake coefficients.

While my supply and uptake functions did not end up in my overall program, I was still able to add tumor cells to the model for future use. They do not currently interact with the simulation in any way. The model starts with 50 tumor cells being placed in one of the corners of the grid. Figure 3.5.2 shows the initial starting positions of the tumor cells. This is the same initial setup of cancer cells and vasculature as seen in the previous models ([11], [16]). The tumor cells start off in the corner and do not move throughout the simulation. Notice that they start off very close to the vasculature, which means they would have enough supply of oxygen, and would not be considered hypoxic. Future work could focus on finishing the uptake function that uses the locations of cancer cells that are now available inside the program.

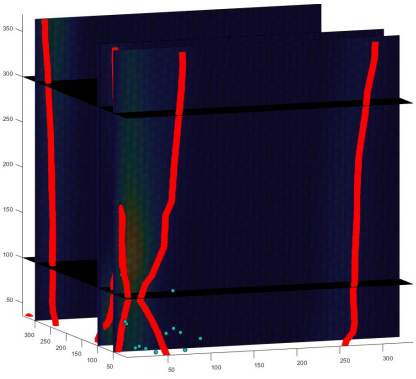

Figure 3.6.3: Cancer Cells With Vasculature

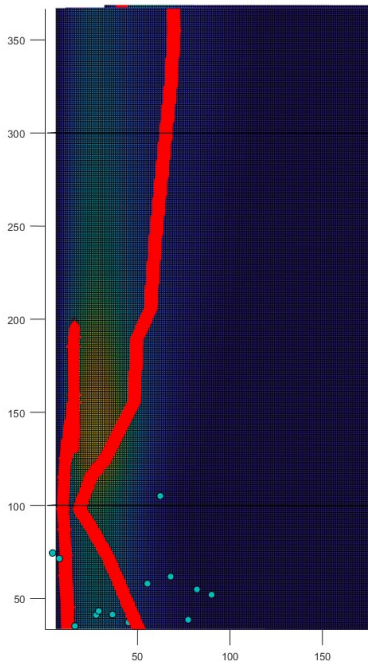

Figure 3.6.4: Cancer Cells With Vasculature

Figures 3.6.3 and 3.6.4 show tumor cells with vasculature and cross-sections of the oxygen concentration matrix zoomed in. The cancer cells are at the bottom of the figures represented by blue circles. The inclusion of cancer cells in the model makes it easier to incorporate updated supply and uptake functions for any future work.

4
Discussion and Conclusions

The goal of this project was to upgrade a previous agent-based model of Triple-Negative Breast Cancer ([11], [16]) by adding a new oxygen diffusion module. The motivation behind adding this module was to have a more realistic way of measuring hypoxia in cancer. The existing model measured the closest Euclidean distance between each cancer cell in the grid and the blood vessels to determine whether the cell was hypoxic. With the addition of my module, we can now have a separate matrix representing the available concentration of diffused oxygen at any given location. That means that we would no longer need to use Euclidean distance between cells and vasculature, and could instead have the cancer cells take up the oxygen concentration available at their location, and become hypoxic if there isn't enough oxygen available.

The vascular diffusion occurred in several steps. First, I wrote a program to diffuse a point source in a 2D grid. The program worked was successful, diffusing equally in every direction in the shape of a circle centered around the point source. The average error between my program and the Python program ([35]) I used to validate my code under 0.4%. This was the simplest diffusion program, so the next thing I did was upgrade the point supply in the grid to a line source. The 2D line diffusion program also worked as intended, with a downward diffusion gradient. The result of the diffusion of the line source visually matched the Python program ([35]) I used to confirm the correctness of my program. The average error was 1.3%, with most

of the error concentrated at the bottom of the grid, just above the boundary. This distribution of error was expected since the finite difference method uses previous rows to calculate the concentrations of oxygen at the next rows of the matrix. Therefore, as we move down the array, the error accumulates from the last row to the next. That concluded the diffusion models in 2-dimensions. Next, I moved up to diffusing a source in three spatial dimensions. I placed a cube source with dimensions $7 \times 7 \times 7$ at the center of the matrix. 7 was an arbitrary side length, but I found it to be a good cube size for visualizing purposes. I tried 3, but it was a bit too small for a $50 \times 50 \times 50$ grid. The 3D diffusion program worked as intended. The cube diffused symmetrically in all directions and matched the gold standard from ([43]), with an average error under 0.3%. With the 2D and 3D test models completed and checked for correctness, I then moved to vascular diffusion of oxygen.

Before I inputted vasculature as the source in my program, I had to make sure the diffusion matched the oxygen diffusion radius range obtained from biology studies and experiments. Therefore, I experimented with my diffusion parameters, α, and the timestep Dt, to obtain a diffusion radius of around 100 microns (50 on the grid, as the grid width is 2 microns). As seen in [9], the normal oxygen diffusion range is 100-200 microns. With the appropriate diffusion parameters set, I then proceeded to input the initial vasculature into the program as starting sources.

In order to diffuse oxygen from blood vessels, I used the code for vasculature from Dr. Norton's previous model ([16]). That code uses a matrix of logical (boolean values) to keep track of locations of vasculature in a $500 \times 500 \times 500$ grid. I used that grid to determine what locations to make my sources in the diffusion model. I did not check for anastomosis, as the initial blood vessels are all assumed to have blood flow. I was able to successfully run the vascular diffusion, and the results look as expected, with oxygen diffusing outward equally in every direction from every vasculature node. The model is now ready to be used in other models for future research. Since I first coded a 2-dimensional diffusion model, my program can also be adjusted to fit in 2D models, similar to the one used in Gevertz et al. ([15]).

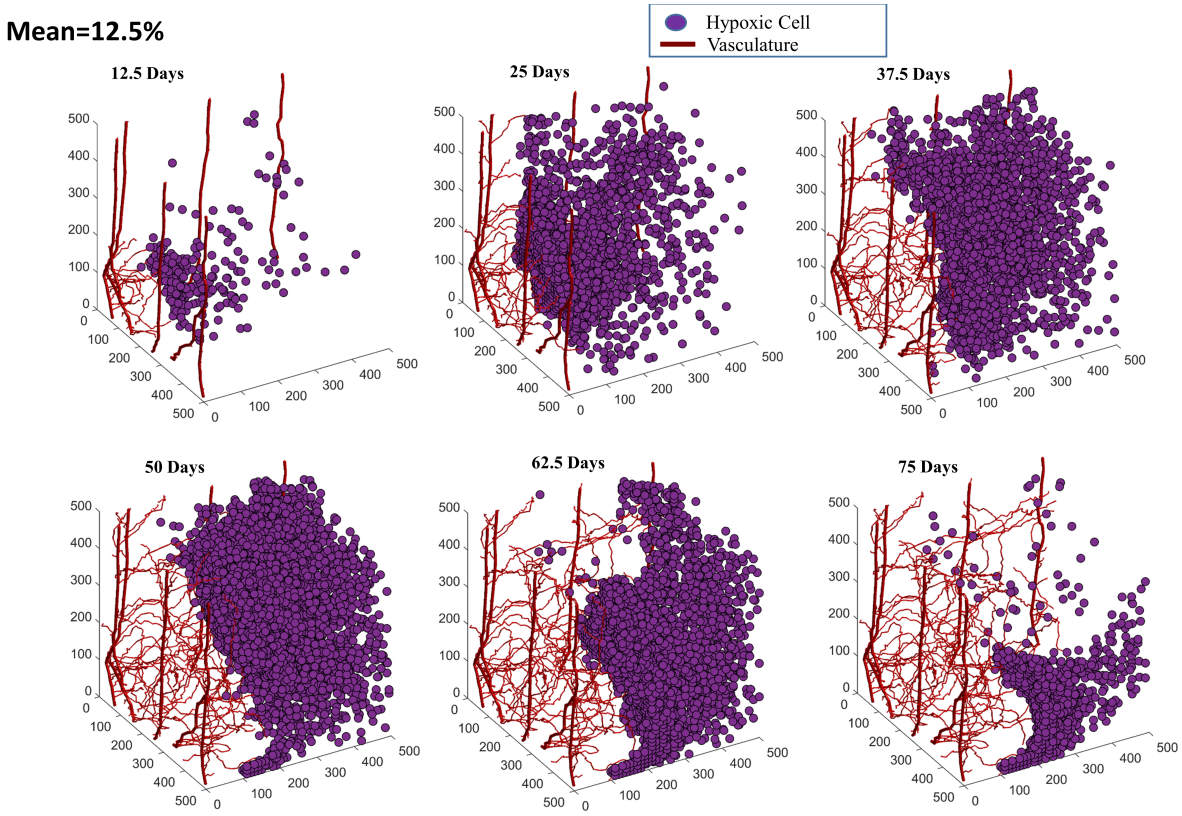

Figure 4.1.1: Hypoxia in Existing Model

4.1 Next Steps

Dr. Norton's previous work includes observing trends of hypoxia in tumors over time. Recall that hypoxic cells are cells that have been deprived of oxygen. Hypoxic cells have been known to have reduced proliferation and migration rates ([28]). Therefore, having an accurate representation of hypoxia in a cancer model would lead to more realistic simulations of cancer dynamics. A longer-term goal for the future of this project would be rerunning some of the older simulations with the new oxygen diffusion model in order to compare hypoxia trends. In the previous models, cancer's access to oxygen has been determined via a straight line distance to the closest vasculature. This time, we would have information about the exact concentration of oxygen available at every grid point.

Figure 4.1.1 shows the hypoxia progression in a previous model developed in Dr. Norton's computational biology lab. The purple cells represent cancer cells that have been deprived of oxygen. As expected, I see that such cells are distributed at a further distance from the vasculature. In theory, the trends should not be drastically different with the new oxygen distribution model. An important aspect of measuring hypoxia with our diffusion model is determining the oxygen threshold for cancer cells. How much oxygen is a negligible amount for hypoxia purposes? How would varying that threshold affect hypoxia trends? Intuitively, the larger the hypoxia threshold for oxygen concentration, the larger the number of hypoxic cells. It would be interesting to perform statistical analysis between the locations of hypoxic cells from the old and new models.

As noted above, the vasculature I diffused in the model has blood flow from the beginning. Meaning, we don't check for sprouting or anastomosis. In the future, it would be interesting to include blood vessels resulting from tumor-induced angiogenesis as oxygen supply. In order to implement that, we would first need to keep track of which blood vessels have undergone anastomosis. Dr. Norton's older model uses another matrix to keep track of that information. Therefore, with a slight modification of my code, we could diffuse oxygen from every vessel with blood flow, and not just the initial 8 capillaries.

Another curious direction one could consider taking this project is the simulation of hyperbaric oxygen therapy. Hyperbaric oxygen therapy refers to increasing the levels of pure oxygen in a patient's body. As seen in [34], HBO (hyperbaric oxygen) has the potential to act as a tumor inhibitor. This could make cancer cells easier to kill with therapies like radiation and chemotherapy. A study found that hyperbaric oxygen therapy had positive effects on men with prostate cancer ([6]). Other studies have shown that hyperbaric oxygen alone has no significant effect on tumor growth and progression ([49], [44]). This model could be a foundation of an ABM that simulates the effect of hyperbaric oxygen therapy on specifically breast cancer. One could investigate how Chimeric Antigen Receptor (CAR) T-cell therapy is affected by hyperbaric oxygen therapy. CAR T-cell therapy has already been simulated in Dr. Norton's lab ([11], [16]),

so adding a hyperbaric oxygen therapy module would be one extra step. Furthermore, after adding an HBO module, that model could be upgraded to a chemotherapy or radiation therapy simulation in triple-negative breast cancer. In other words, this project can be expanded to incorporate HBO to test the in-silico effectiveness of chemotherapy and/or radiation therapy.

There are some interesting mathematical future steps this project could incorporate. Namely, one could bound the roundoff error of the program, which occurs due to the approximations used in our numerical evaluations. The width of the mesh grid we use for our numerical evaluation plays a big role in the accuracy of the program. The smaller Dx, Dy, and Dt are, the more accurate the results will be. This is also seen in the Mathematics Background section where we approximated U_x and U_t. The smaller the grid sizes, the closer the approximations are to the actual partial derivatives. However, if we make the grid size too small, that exponentially slows down the program. So, we are dealing with roundoff and computation errors when we pick the time step and grid width. A future addition to the project could focus on writing an error analysis function and try to put bounds on the error function.

Another future mathematical direction for this project is to upgrade the numerical evaluation method. As mentioned in the Mathematics Background Section, my program incorporates the Explicit Finite Difference Method. Instead, one could use the Implicit Finite Difference Method, namely, the Crank-Nicolson method, which incorporates linear algebra to approximate the solutions to a PDE. Since my code is written in Matlab, implementing an FDM that's based on matrix multiplication could potentially make the code considerably faster, as Matlab is very efficient with its linear algebra operations.

4.2 Personal Reflection

Working on a senior project was a journey full of curiosity, learning, exploring, creativity, and problem-solving. I started off my senior year not even having seen a partial differential equation. Within a few months, I managed to learn now only what PDEs are, but how to apply them to biology, and how to evaluate them numerically. It was the first time I had to teach myself

an entire math topic from scratch. The math was so enticing, that I even took an additional course in PDEs to learn even more. I learned not only a new math field but strategies to self-teach. Additionally, I was able to build on a model I worked on during Bard Summer Research Institute in 2021. It felt rewarding going back to what (my colleagues and) I started years ago. Throughout the course of the past eight months, I spent every week designing the next steps of the project, setting weekly goals, reassessing progress and the big picture, and course-correcting. Of course, I ran out of time and did not get to fully accomplish my very last desired result, but watching myself make some progress every week was truly a growing experience. Now, I know how to work with PDEs both analytically and numerically, I know the trends and specifics of diffusion, and I know about cancer, our vasculature, agent-based and hybrid modeling, and code vectorization. It feels incredible knowing how much more I know now compared to my first day of senior year purely due to my senior project.

Appendix A
Code for 2D/3D Diffusion, Supply, and Sink

A.1 Code for 2D Diffusion

```
%This is the code for diffusing a line at the top of a 2D grid of
   size 50. We set Dx, Dy, Dt, alpha, and then diffuse the line
   source until the aggregate change in he matrix from one time
   step to the next is less than 0.1% of the source value (100).
   This Code is not vectorized.
size = 50;
Dx=1; % step size
Dy=1;


alpha=5; % diffusion constant
global U
U=[];



%--BCs--
U(1:size,1:size) = 0 ;
U(1,1:size) = 0;
U(size,1:size) = 0;
U(1:size,1) =0;
U(1:size,size) = 0;


%--IC's
for i=2:49
    U(1,i)=100;
end
maxx=max(max(U));
```

```
%
   ------------------------------------------------------------------------

%calculate timestep
%Dt = Dx^2/(2*alpha^2);

%end simulation after 20,000 iterations (steadystate)
max_iter=20000;
%---set up the formulas
   ---------------------------------------------
iter=0;
check=1;

while check==1
   difference=0;
   U_old = U;
for i = 2:N-1
for j = 2:N-1
%Finie Difference Formula for 2D
   new=Dt*((U_old(i+1,j)-2*U_old(i,j)+U_old(i-1,j))/Dx^2 ...
                      + (U_old(i,j+1)-2*U_old(i,j)+U_old(i,j-1))/
                         Dy^2) ...
                      + U_old(i,j);
    %count aggregate difference between one iteration and the next
   difference = difference+ abs(new-U(i,j))
end
end
%Make sure the difference is less than 0.1% of the starting source
   . Can be arbitrary
if(difference>=0.001*maxx)
    iter=iter+1;

 %--If 20,000 time steps aren't enough, we end early
    -------------------------

    if(iter>maxx)
    check=0;
    disp("terminated early");
    end

 %--Else, if the error is smaller than 0.1%, we finish the program
    ........................

else
    check=0;
    disp("Done!")
end
end
```

```
heatmap(U)
```

A.2 For Loop for 3D Diffusion

```
    %Most of the code stays the same when we move up to 3
       Dimensions. U now changes to a 50x50x50 matrix, and we also
        change BCs to include all 6 boundary planes, and ICs to
       include three dimensions.

%Finie Difference Formula for 3D
check =1;

while check ==1

for i = 2:size-1
for j = 2:size-1
for k = 2:size-1
   new=(Dt*alpha*((U_old(i+1,j,k)-2*U_old(i,j,k)+U_old(i-1,j,k))/
      Dx^2 ...
                    + (U_old(i,j+1,k)-2*U_old(i,j,k)+U_old(i,j
                       -1,k))/Dy^2 ...
                    + (U_old(i,j,k+1)-2*U_old(i,j,k)+U_old(i,j,k
                       -1))/Dz^2) ...
                    + U_old(i,j,k));
    %count aggregate difference between one iteration and the next
   difference = difference+ abs(new-U(i,j,k))
end
end
end
%Make sure the difference is less than 0.1% of the starting source
   . Can be arbitrary
if(difference>=0.001*maxx)
    iter=iter+1;

 %--Is 20,000 time steps isn't enough, we end early
    ------------------------

    if(iter>maxx)
    check=0;
    disp("terminated early");
    end

 %--Else, if the error is smaller than 0.1%, we finish the program
    ........................
```

```
else
    check=0;
    disp("Done!")
end %end if/else statement
end %end while loop
```

A.3 Vectorized Version

```
% This is the vectorized version of the double/triple four loops
   seen above. Notice that here, we do not have any looping
   through i,j, or k. The only loop we use is a while loop to
   check for a steady-state

i = 2:N-1;
j = 2:N-1;
k = 2:N-1;
difference=10; %arbitrary large number
U_n=U; % Placeholder matrix to calculate new values each iteration
tic

%maximum error can't be more than a tiny percentage of the highest
    temperature

while difference>= maxx*0.00001  %can increase the tolerated
   percent change as needed
   U_n(i,j,k)=Dt*alpha*((U(i+1,j,k)-2*U(i,j,k)+U(i-1,j,k))/Dx^2
      ...
                      + (U(i,j+1,k)-2*U(i,j,k)+U(i,j-1,k))/Dy^2
                        ...
                      + (U(i,j,k+1)-2*U(i,j,k)+U(i,j,k-1))/Dz^2)
                        ...
                      + U(i,j,k) ;

 difference=(max(max(max(abs(U_n-U)))));
  U(i,j,k)=U_n(i,j,k);
end
toc
```

A.4 Diffusing From Vasculature

```
%When we want to change our program to diffuse oxygen from the
   vasculature, the only aspect we need to modify is the initial
   conditions. Now, instead of placing the sources at the top
   boundary or in the middle of the grid, we check for locations
```

```
   that have vasculature in them. In other words, every point in
   the grid that has a vasculature node in it starts off as a
   source.

%Changed IC: voxelgrid contains three different matrices. The one
   we need is called Agent, which is a 50x50 logical array. If the
    initial grid has a vasculature source in it, the corresponding
    value in Agent is going to be 1, otherwise, it will be 0. find
   (voxelgrid.Agent==1) returns the indices of all gridpoints that
    contain vasculature. Note that I do NOT create vasculature in
   this program: it is adapted from Dr. Norton's existing code.
   The initial vasculature can be seen in the Results section.


U(find(voxelgrid.Agent==1))=100;

%BCs remain the same
U(1, 1:N, 1:N) = 0;
U(N,1:N, 1:N) = 0;
U(1:N, 1:N,1) =0;
U(1:N, 1:N,N) =0;
U(1:N,1,1:N) = 0;
U(1:N, N,1:N) =0;
```

A.5 Supply Function

```
%The function below takes in a 3d location and returns an integer
   value of how many oxygen sources it neighbors. voxelgrid.Agent
   is a boolean matrix with 1's at locations of vasculature and 0'
   s in locations with no vasculature. We calculate supply at a
   point by checking all its 27 neighbors in 3D (and itself) and
   counting how many vasculature nodes it neighbors.

function[s]=supply3d(point)
global voxelgrid
x=point(1); y=point(2); z=point(3);
%calculate neighboring coordinates
x_left=max(1, x-1); x_right=min(50, x+1); y_top=min(50, y+1);
   y_bot=max(1, y-1); z_up=min(50, z+1); z_down=max(1, z-1);
%find all neighbors, including itself
cords={ [x_left y z], [x_right y z], [x y_top z], [x y_bot z], [
   x_right y_top z], [x_right y_bot z], [x_left y_top z], [x_left
   y_bot z]...
    ,[x y z_up], [x y z_down], [x_right y z_up], [x_right y z_down
       ], [x_left y z_up], [x_left y z_down], ....
```

```
    [x_right y_top z_up] [x_right y_top z_down] [x_right y_bot
       z_up], [x_right y_bot z_down],...
    [x_left y_top z_up], [x_left y_top z_down], [x_left y_bot z_up
       ], [x_left y_bot z_down]...
    , [x y_top z_up], [x y_top z_down], [x, y_bot z_up] [x, y_bot
       z_down], [x y z]
    };
%n is an array of 1s and 0s representing whether its neighboring
   locations have vasculature in them
n=zeros(length(cords));
for ind =1:length(cords)
    loc=cords{ind};
    n(ind)=voxelgrid.Agent(loc(1), loc(2), loc(3));
end

s=0;
%sum up all of n to find the final supply number
for i=1:length(n)
    s=s+n(i);
end

end
```

A.6 Uptake Function

```
%This function takes in a location and calculates its uptake value
    by counting how many cancer cells that location is close to.
   XYZ is a matrix containing the locations of all the cancer
   cells.

function[intake]=uptake(point)
global XYZ
%variable to control sensing radius. Currently set at 3.
global sensingR
cells=XYZ;
intake=0;
%Loop through the cells
for ind=1:length(cells)
%check if the distance from each cell to our location is less than
    the specified radius
    if pdist2(point, cells(ind,:)) < sensingR
    %if within the sensing radius, we add 1 to the uptake count
        change=1;
    else
    %otherwise, we add zero
        change=0;
```

```
    end

    intake=intake+change;
end
end
```

Bibliography

[1] The Editors of Encyclopaedia Britannica, *Diffusion*, 2022, `https://www.britannica.com/science/diffusion.`.

[2] Steve Brunton, *Deriving the Heat Equation: A Parabolic Partial Differential Equation for Heat Energy Conservation*, Youtube, 2022.

[3] Steve Brunton, *Deriving the Heat Equation in 2D and 3D (and in N Dimensions!) with Control Volumes and Vector Calculus*, Youtube, 2022.

[4] Stefano Casarin and Eleonora Dondossola, *An agent-based model of prostate Cancer bone metastasis progression and response to Radium223*, BMC cancer **20** (2020), 1–19.

[5] JR Chasnov, *Differential Equations for Engineers*, Lecture notes for Coursera. The Hong Kong University of Science and Technology publishing. (2019).

[6] Marc A Dall'Era, Neil B Hampson, R Alex Hsi, Berit Madsen, and John M Corman, *Hyperbaric oxygen therapy for radiation induced proctopathy in men treated for prostate cancer*, The Journal of urology **176** (2006), no. 1, 87–90.

[7] David Dingli, Matthew D Cascino, Krešimir Josić, Stephen J Russell, and Željko Bajzer, *Mathematical modeling of cancer radiovirotherapy*, Mathematical biosciences **199** (2006), no. 1, 55–78.

[8] Fabien Dournac, *HMPI Parallelization for numerically solving the 3D Heat equation*, 2003.

[9] Katherine L. Eales, Kate E.R. ollinshead, and Daniel A Tennant, *Hypoxia and metabolic adaptation of cancer cells*, Oncogenesis **5** (2016), no. 1, e190–e190.

[10] Stanley J Farlow, *Partial differential equations for scientists and engineers*, Courier Corporation, 1993.

[11] Henning Fischel, Tina Giorgadze, Ansel Tessier, and Kerri-Ann Norton, *Computational modeling of chimeric antigen receptor (car) t-cell therapy of a binary model of antigen receptors in breast cancer*, 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2021, pp. 3267–3274.

[12] J. Folkman and R. Kalluri, *Cancer without disease.*, Nature **427, 787** (2004), DOI https://doi.org/10.1038/427787a.

[13] Monica Fornier and Pierre Fumoleau, *The paradox of triple negative breast cancer: novel approaches to treatment*, The breast journal **18** (2012), no. 1, 41–51.

[14] Allan J Franko and Robert M Sutherland, *Oxygen diffusion distance and development of necrosis in multicell spheroids*, Radiation Research **79** (1979), no. 3, 439–453.

[15] Jana Gevertz, Zahra Aminzare, Kerri-Ann Norton, Judith Pérez-Velázquez, Alexandria Volkening, and Katarzyna A Rejniak, *Emergence of anti-cancer drug resistance: exploring the importance of the microenvironmental niche via a spatial model*, Applications of dynamical systems in biology and medicine, 2015, pp. 1–34.

[16] Tina Giorgadze, Henning Fischel, Ansel Tessier, and Kerri-Ann Norton, *Investigating Two Modes of Cancer-Associated Antigen Heterogeneity in an Agent-Based Model of Chimeric Antigen Receptor T-Cell Therapy*, Cells **11** (2022), no. 19, 3165.

[17] Chang Gong, Oleg Milberg, Bing Wang, Paolo Vicini, Rajesh Narwal, Lorin Roskos, and Aleksander S Popel, *A computational multiscale agent-based model for simulating spatio-temporal tumour immune response to PD1 and PDL1 inhibition*, Journal of the Royal Society Interface **14** (2017), no. 134, 20170320.

[18] Florian R Greten and Sergei I Grivennikov, *Inflammation and cancer: triggers, mechanisms, and consequences*, Immunity **51** (2019), no. 1, 27–41.

[19] Richard Haberman, *Applied partial differential equations with Fourier series and boundary value problems*, Pearson Higher Ed, 2012.

[20] Douglas Hanahan and Robert A Weinberg, *Hallmarks of cancer: the next generation*, cell **144** (2011), no. 5, 646–674.

[21] Zarifeh Heidary, Jafar Ghaisari, Shiva Moein, and Shaghayegh Haghjooy Javanmard, *The double-edged sword role of fibroblasts in the interaction with cancer cells; an agent-based modeling approach*, PloS one **15** (2020), no. 5, e0232965.

[22] Victor W. M. van Hinsbergh, *Angiogenesis: Basics of Vascular Biology*, Vascularization for Tissue Engineering and Regenerative Medicine (Wolfgang Holnthoner, Andrea Banfi, James Kirkpatrick, and Heinz Redl, eds.), Springer International Publishing, Cham, 2017, pp. 1–29.

[23] Klaus A Hoffman and Steve T Chiang, *Computational fluid dynamics*, Engineering Education System **2** (2000).

[24] Emilia Kozłowska, Rafał Suwiński, Monika Giglok, Andrzej Świerniak, and Marek Kimmel, *Mathematical model predicts response to chemotherapy in advanced non-resectable non-small cell lung cancer patients treated with platinum-based doublet*, PLoS Computational Biology **16** (2020), no. 10, e1008234.

[25] Hans Petter Langtangen, *Finite difference methods for diffusion processes*, University of Oslo (2013).

[26] Mengfei Li, Patricia Perez-Calleja, Bumkyu Kim, Cristian Picioreanu, and Robert Nerenberg, *Unique stratification of biofilm density in heterotrophic membrane-aerated biofilms: An experimental and modeling study*, Chemosphere (2023), 138501.

[27] AI Liapis, GG Lipscomb, Orrin K Crosser, and E Tsiroyianni-Liapis, *A model of oxygen diffusion in absorbing tissue*, Mathematical modelling **3** (1982), no. 1, 83–92.

[28] Sensen Lin, Shuying Wan, Li Sun, Jialiang Hu, Dongdong Fang, Renping Zhao, Shengtao Yuan, and Luyong Zhang, *Chemokine C-C motif receptor 5 and C-C motif ligand 5 promote cancer cell migration under hypoxia*, Cancer Science **103** (2012), no. 5, 904–912.

[29] J David Logan, *Applied partial differential equations*, Springer, 2014.

[30] Jeremy S Logue and Deborah K Morrison, *Complexity in the signaling network: insights from the use of targeted inhibitors in cancer therapy*, Genes & development **26** (2012), no. 7, 641–650.

[31] Roberta Lugano, Mohanra Ramachandran, and Anna Dimberg, *Tumor angiogenesis: causes, consequences, challenges and opportunities*, Cellular and Molecular Life Sciences **77** (2020), 1745–1770.

[32] Romina Martin, *Combining system dynamics and agent-based modeling to analyze social-ecological interactions—an example from modeling restoration of a shallow lake*, `https://www.frontiersin.org/articles/10.3389/fenvs.2015.00066/full`.

[33] *MATLAB version 9.9.0.123456 (R2020b)*, The Mathworks, Inc., Natick, Massachusetts, 2021.

[34] Ingrid Moen and Linda EB Stuhr, *Hyperbaric oxygen therapy and cancer—a review*, Targeted oncology **7** (2012), 233–242.

[35] G Nervadof, *Solving 2D Heat Equation Numerically using Python*, `https://levelup.gitconnected.com/solving-2d-heat-equation-numerically-using-python-3334004aa01a`.

[36] New World Encyclopedia, *Heat conduction — New World Encyclopedia,* 2017.

[37] Kerri-Ann Norton, Kideok Jin, and Aleksander S Popel, *Modeling triple-negative breast cancer heterogeneity: Effects of stromal macrophages, fibroblasts and tumor vasculature*, Journal of theoretical biology **452** (2018), 56–68.

[38] Kerri-Ann Norton, Travis Wallace, Niranjan B Pandey, and Aleksander S Popel, *An agent-based model of triple-negative breast cancer: the interplay between chemokine receptor CCR5 expression, cancer stem cells, and hypoxia*, BMC systems biology **11** (2017), no. 1, 1–15.

[39] Kerri-Ann Norton and Aleksander S Popel, *Effects of endothelial cell proliferation and migration rates in a computational model of sprouting angiogenesis*, Scientific reports **6** (2016), no. 1, 1–10.

[40] Kerri-Ann Norton and Aleksander S Popel, *An agent-based model of cancer stem cell initiated avascular tumour growth and metastasis: the effect of seeding frequency and location*, Journal of The Royal Society Interface **11** (2014), no. 100, 20140640.

[41] Sarah A Nowak, Andrew Parker, Archana Radhakrishnan, Nancy Schoenborn, and Craig Evan Pollack, *Using an agent-based model to examine de-implementation of breast cancer screening*, Medical care **59** (2021), no. 1, e1.

[42] Megan M Olsen and Hava T Siegelmann, *Multiscale agent-based model of tumor angiogenesis*, Procedia Computer Science **18** (2013), 1016–1025.

[43] Alex Pedcenko, *3D Heat equation solution with FD in MATLAB*, 2023, `https://github.com/aa3025/heat3d/releases/tag/1.0.5`.

[44] Angela M Poff, Csilla Ari, Thomas N Seyfried, and Dominic P D'Agostino, *The ketogenic diet and hyperbaric oxygen therapy prolong survival in mice with systemic metastatic cancer*, PloS one **8** (2013), no. 6, e65522.

[45] Guillem Pratx and Daniel S Kapp, *A computational model of radiolytic oxygen depletion during FLASH irradiation and its effect on the oxygen enhancement ratio*, Physics in Medicine & Biology **64** (2019), no. 18, 185005.

[46] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical recipes in C*, Second, Cambridge University Press, Cambridge, 1992. The art of scientific computing.

[47] Stephen Riffle and Rashmi S Hegde, *Modeling tumor cell adaptations to hypoxia in multicellular tumor spheroids*, Journal of Experimental & Clinical Cancer Research **36** (2017), 1–10.

[48] Melanie Rivera, Leslie Toledo-Jacobo, Elsa Romero, Tudor I Oprea, Melanie E Moses, Angela Wandinger-Ness, and Martha M Grimes, *Agent-based modeling predicts RAC1 is critical for ovarian cancer metastasis*, Molecular Biology of the Cell **33** (2022), no. 14, ar138.

[49] Pieter J Schoen, Gerry M Raghoebar, Jelte Bouma, Harry Reintsema, Arjan Vissink, Wouter Sterk, and Jan LN Roodenburg, *Rehabilitation of oral function in head and neck cancer patients after radiotherapy with implant-retained dentures: effects of hyperbaric oxygen therapy*, Oral oncology **43** (2007), no. 4, 379–388.

[50] David S Schultz and William E King, *On the analysis of oxygen diffusion and reaction in biological systems*, Mathematical biosciences **83** (1987), no. 2, 179–190.

[51] Imtiaz Siddiqui, Vanna Sanna, Nihal Ahmad, Mario Sechi, and Hasan Mukhtar, *Resveratrol nanoformulation for cancer prevention and therapy*, Annals of the New York Academy of Sciences **1348** (2015), DOI 10.1111/nyas.12811.

[52] Matthew J Simpson and Adam J Ellery, *An analytical solution for diffusion and nonlinear uptake of oxygen in a spherical cell*, Applied Mathematical Modelling **36** (2012), no. 7, 3329–3334.

[53] Jonathan W Song et al., *Anastomosis of endothelial sprouts forms new vessels in a tissue analogue of angiogenesis.*, Integrative biology : quantitative biosciences from nano to macro **4,8** (2012), 857-62.

[54] Angéliqu Stéphanou, Anne-Cécile Lesart, J Deverchère, A Juhem, A Popov, and F Estève, *How tumour-induced vascular changes alter angiogenesis: insights from a computational model*, Journal of theoretical biology **419** (2017), 211–226.

[55] Ian F Tannock, *Oxygen diffusion and the distribution of cellular radiosensitivity in tumours*, The British journal of radiology **45** (1972), no. 535, 515–524.

[56] Michael Ventoso and Kerri-Ann Norton, *Simulating an immune response with a combined agent-based model of a triple-negative breast cancer tumor and vascular network*, 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2020, pp. 1303–1310.

[57] Robert A Weinberg, *The biology of cancer*, Garland science, 2013.

[58] WPTM Wickramaarachchi and SSN Perera, *An SIER model to estimate optimal transmission rate and initial parameters of COVD-19 dynamic in Sri Lanka*, Alexandria Engineering Journal **60** (2021), no. 1, 1557–1563.

[59] Franziska et al. van Zijl, *"Initial steps of metastasis: cell invasion and endothelial transmigration."*, Mutation research **vol 728,1-2** (2011), 23-34, DOI doi:10.1016/j.mrrev.2011.05.002.