Honors Thesis

Department of Mathematics

University of Colorado Boulder

April 2, 2024

Numerical methods for a PDE system modeling tumor angiogenesis

Graham Mauer

Thesis Advisor: Dr. Padi Fuster Aguilera, Department of Mathematics

Outside Reader: Dr. Eduardo Corona, Department of Applied Mathematics

Honors Council Representative: Dr. Magdalena Czubak, Department of Mathematics

Abstract

This thesis is centered on the foundation needed for performing numerical simulations of a partial differential system (PDE) of reaction-diffusion equations modeling tumor growth in two dimensions. It includes the derivation of a particular PDE system modeling *chemotaxis* with a generalized logistic growth of the cell population density, and original results on the one-dimensional case. In particular, we show the effect of the generalized logistic growth on the population density of the organism. The second part of this thesis lays down the foundation needed to extend the numerical simulations to a two-dimensional Euclidean framework, confirming the accuracy of results through various numerical and analytical methods. The results of this work may pave the way to a deeper understanding of tumor angiogenesis over curved organ surfaces.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION TO THE BIOLOGICAL PROBLEM

Organisms respond to stimuli to survive, for example animals foraging for food or sperm finding eggs during fertilization. The movement of organisms in response to a chemical stimuli is called *chemotaxis* [1], stemming from the Greek root *-taxis* meaning movement. Specifically, it involves the directed movement of cells or organisms in response to chemical gradients or signals in their environment. Chemotaxis is ubiquitous throughout biology and ecology, appearing across both macro and micro scales, such as bee pollination [2] or bacterium finding food [3].

Chemotaxis is the driving mechanism behind tumor angiogenesis [4], where tumors hijack existent nearby blood vessels in order to secure nutrients needed to grow. During tumor angiogenesis, the tumor releases a specific chemical, tumor angiogenesis factor (TAF), into the bloodstream. Then, molecular signals induce neighboring blood vessel cells to form new vascular pathways, ensuring the flow of essential nutrients and oxygen to the expanding tumor [5]. This influx of nutrients is necessary for the tumor to grow larger than two to three millimeters [6]. See figure 1.1.

When endothelial cells, which make up the blood vessel walls, sense the TAF, they react according to the Weber-Fechner law [7], which states that the sensitivity of the endothelial cells to TAF is of logarithmic order. This ensures that even with a high concentration of TAF, the endothelial cells can determine the direction of the TAF gradient. In addition, we assume that the movement timescale of TAF in the cellular matrix and the lifespan of endothelial cells are comparable [8], i.e., endothelial cells can reproduce at comparable timescale to their relocation. Thus, we include logistic growth for the population density, which results in the species converging to the carrying capacity determined by the local environment.

Such cellular behavior is represented by a class of partial differential equations (PDEs) called *reaction-diffusion* equations and can be extended to model other events, such as rumor propagation [9], wealth concentration [10], and the spread of disease. Working with reaction-diffusion

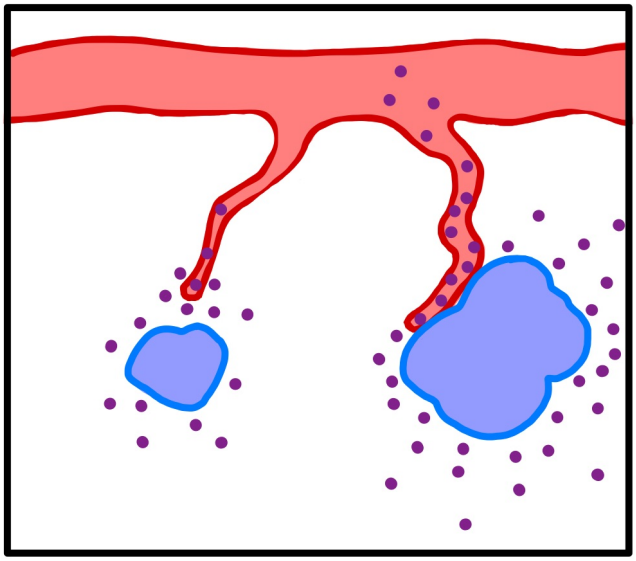

Figure 1.1: Tumors (blue) release TAF (purple) into the cellular matrix and blood vessels (red) create new pathways to the TAF source.

equations modeling chemotaxis could open up new avenues of approach in these different fields.

A PDE model for chemotaxis was first developed by Patlak [11] in the 1950's and refined by Keller and Segel in the 1970's [12]. In the last 30 years, many variations of the Keller-Segel model have been examined which highlight the importance of chemotaxis in biology and medicine. However, the volume of literature considerably shrinks for this particular model, especially with assumptions such as logarithmic sensitivity and logistic growth for the endothelial cells. The one-dimensional case has been studied both analytically and numerically by Fuster in [13], but to our knowledge no literature has been published examining numerical solutions to this system in the case of the generalized logistic growth in one or higher dimensions. This thesis covers the author's original results on the characterization of the generalized logistic growth of the population density in the 1D case, as well as developing numerical machinery to extend these results to 2D Euclidean spaces.

CHAPTER 2

DERIVATION OF THE MODEL

Chemotaxis can be described by a reaction-diffusion model. A general reaction-diffusion framework includes diffusion, which causes the spread of the organism from higher density areas to lower density areas, and some reaction from the organism to, i.e., a certain chemical.

2.1 Derivation of the Model

Here, $\hat{u}$ and $\hat{c}$ are scalar functions representing the population density of the blood vessel cells and the concentration of the secreted TAF chemical, respectively. The time evolution of the population density of endothelial cells, $\hat{u}$, is based on a conservation over an infinitesimal unit volume, Ω. We expect the number of cells in the volume to only change by the flux of cells out, Φ, through the boundary of the volume, $\partial\Omega$, and the spontaneous birth or death of cells within Ω, modeled by $f(\hat{u}, \hat{c})$. Thus, a conservation law arises:

$$\frac{\partial}{\partial t}\int_{\Omega} \hat{u}(x,t)dx = -\int_{\partial\Omega} \Phi \cdot \hat{n}\, dS + \int_{\Omega} f(\hat{u}, \hat{c}). \tag{2.1}$$

The flux, Φ through the boundary $\partial\Omega$, includes both diffusive and chemotactic fluxes through the boundary: $\Phi = \Phi_{\text{diff}} + \Phi_{\text{chemo}}$. Φ_{diff} obeys Fick's Law, which states a substance will diffuse proportionally to the gradient of its concentration, $\Phi_{\text{diff}} = -\nabla D\hat{u}$ with $D \in \mathbb{R}^{+}$. Φ_{chemo} is the flux from chemotaxis, $\Phi_{\text{chemo}} = \chi(\hat{u}, \hat{c})\nabla\hat{c}$ where χ is a function describing how $\hat{u}$ is transported in respect to the gradient of the chemical concentration. Then (2.1) becomes:

$$\frac{\partial}{\partial t}\int_{\Omega} \hat{u}(x,t)dx = -\int_{\partial\Omega} (-\nabla D\hat{u} + \chi(\hat{u}, \hat{c})\nabla\hat{c}) \cdot \hat{n}\, dS + \int_{\Omega} f(\hat{u}, \hat{c})dx. \tag{2.2}$$

Applying the Divergence Theorem and passing the time derivative into the integral on the left hand side, we obtain

$$\int_\Omega \frac{\partial}{\partial t}\hat{u}(x,t)dx = -\int_\Omega (\nabla \cdot (-D\nabla\hat{u} + \chi(\hat{u},\hat{c})\nabla\hat{c}) + f(\hat{u},\hat{c}))dx. \tag{2.3}$$

We now obtain the PDE describing the evolution of the organism population

$$\frac{\partial}{\partial t}\hat{u}(x,t) = D\Delta\hat{u} - \nabla \cdot (\chi(\hat{u},\hat{c})\nabla\hat{c}) + f(\hat{u},\hat{c}). \tag{2.4}$$

Similarly, for the chemical concentration, $\hat{c}$, we obtain

$$\frac{\partial}{\partial t}\int_\Omega \hat{c}(x,t)dx = -\int_{\partial\Omega} \Psi \cdot \hat{n}\, dS + \int_\Omega g(\hat{u},\hat{c})dx. \tag{2.5}$$

Here, $\Psi = \Psi_{\text{diff}} = -\nabla\varepsilon\hat{c}$ is the flux of $\hat{c}$ through the boundary due to diffusion. Moreover, $g(\hat{u},\hat{c})$ describes the creation or destruction of the chemical in Ω. We now can write the whole system as

$$\begin{aligned} \frac{\partial}{\partial t}\hat{u} &= -\nabla \cdot (-D\nabla\hat{u} + \chi(\hat{u},\hat{c})\nabla\hat{c}) + f(\hat{u},\hat{c}), \\ \frac{\partial}{\partial t}\hat{c} &= -\nabla \cdot (-\varepsilon\nabla\hat{u}) + g(\hat{u},\hat{c}). \end{aligned} \tag{2.6}$$

In this thesis we examine a particular version of (2.6) with logarithmic chemotactic sensitivity and generalized logistic growth for the chemical concentration. More specifically:

$$\chi(\hat{u},\hat{c}) = \chi\frac{\hat{u}}{\hat{c}}, \quad f(\hat{u},\hat{c}) = \kappa_1\hat{u}\left[1 - \left(\frac{\hat{u}}{\kappa_2}\right)^\alpha\right], \quad g(\hat{u},\hat{c}) = -\mu\hat{u}\hat{c} - \sigma\hat{c}. \tag{2.7}$$

Above, $\kappa_1, \kappa_2, \alpha \in \mathbb{R}^+$ and $\chi, \mu, \sigma \in \mathbb{R}$. $\chi > 0$ describes an attractive system and $\chi < 0$ describes a repulsive system. $|\chi|$ is a measure of the strength of the chemotactic sensitivity. This choice of parameters in (2.7) leads to logarithmic sensitivity as $\chi\frac{\hat{u}}{\hat{c}} = \chi\hat{u}\nabla\ln\hat{c}$. We choose $g(\hat{u},\hat{c})$ such that it has an interaction term between $\hat{u}$ and $\hat{c}$, as well as including the decay of $\hat{c}$. This choice of (2.7) leads to possible analytical and numerical difficulties as if $\hat{c}$ is small, $|\ln\hat{c}|$ is large. For this reason,

we use a Hopf-Cole transformation to avoid possible singularities. Notice that the usual logistic growth is when $\alpha = 1$, and we define *generalized logistic growth* to be when $\alpha \in \mathbb{R}^+$.

2.2 Transforming the 1D Model

We transform the model to remove the aforementioned singularities caused by the logarithmic term. We apply the transformations, with the last transformation being the Hopf-Cole transformation. We also drop the '*hat*' notation on $\hat{u}$ and $\hat{c}$ in favor of u and c for transformed variables:

$$c = e^{\sigma t}\hat{c}, \qquad u = \frac{\hat{u}}{\kappa_2}, \qquad v = \frac{\nabla c}{c}. \tag{2.8}$$

The system becomes

$$\begin{aligned} \frac{\partial}{\partial t}u &= -\chi\nabla\cdot(uv) + D\Delta u + \kappa_1\kappa_2 u(1-u^\alpha), \\ \frac{\partial}{\partial t}v &= -\nabla(\mu\kappa_2 u - \varepsilon|v|^2). \end{aligned} \tag{2.9}$$

Notice that the second equation is in fact a vectorial equation, given that now v represents the proportional rate of change of the chemical concentration. In the next section, do the non-dimensional analysis of these equations in 1D scalar case, but one can obtain in a similar manner the non-dimensional equations on higher dimensions.

2.3 Non-dimensionalizing the Model

Non-dimensionalizing a model allows for the representation of the system in a way not dictated by units, which can reveal scales intrinsic to the system. The first step is to examine the units of each term in (2.9). M, L, and T correspond to mass, length, and time respectively and α is a non-dimensional constant. Let

$$[\hat{u}] = \frac{M}{L}, \quad [u] = \frac{M}{L}, \quad [\hat{c}] = \frac{M}{L}, \quad [\mu] = \frac{L}{MT}, \quad [\sigma] = \frac{1}{T},$$

$$[D] = \frac{L^2}{T}, \quad [\varepsilon] = \frac{L^2}{T}, \quad [\chi] = \frac{L^2}{T}, \quad [\kappa_1] = \frac{1}{T}, \quad [\kappa_2] = \frac{M}{L}.$$

Using the rescalings

$$t \to \frac{\chi\mu\kappa_2}{D}t, \quad x \to \frac{\sqrt{\chi\mu\kappa_2}}{D}x, \quad v \to \text{sign}(\chi)\sqrt{\frac{\chi}{\mu\kappa_2}}v,$$

we obtain

$$\begin{aligned} \frac{\partial}{\partial t}u &= -\nabla \cdot (uv) + \Delta u + ru(1-u^\alpha), \\ \frac{\partial}{\partial t}v &= -\nabla\left(u - \frac{\varepsilon}{\chi}|v|^2\right) + \frac{\varepsilon}{D}\Delta v. \end{aligned} \tag{2.10}$$

Above, $r = \frac{\kappa_1 D}{\chi\mu\kappa_2}$. We also make the assumption that $\chi = D = 1$ and $\varepsilon \ll 1$ [14]. This non-dimensionalized form will allow us to use a combination of finite difference methods to evolve this system through time in 1D.

2.4 Numerical Strategy

Throughout this thesis we will use finite difference methods to build a model for solving equations (2.10). Finite differences are a method of approximating the derivatives of a function and are further explained in section 4.1. The goal of this thesis is to build the numerical simulation framework for a particular coupled non-linear system of PDEs, establishing a strong foundation for future numerical work on (2.10) in 2D. The foundation we construct here can then be easily extended to add both the chemotaxis flux and the evolution of the chemical concentration. As (2.10) is a hyperbolic-parabolic model when $\chi\mu > 0$, we begin with the simplest case: numerical solutions of the 1D advection-diffusion equation. We then extend our results to the 2D advection-diffusion equation.

One challenge in the 2D model lies in the numerical implementation of the $\nabla \cdot (uv)$ term in (2.9). Here v is a vector and we expand $\nabla \cdot (uv)$ as

$$\nabla \cdot (uv) = u\nabla \cdot v + v \cdot \nabla u.$$

Then we can write the population density equation of (2.9) as

$$u_t = D\Delta u + v \cdot \nabla u + \eta(u\nabla \cdot v, u^m, ...). \tag{2.11}$$

Notice that in (2.9), we have $u_t = h(v, \nabla v, u, \nabla u, uv, ...)$. A suitable first step towards building the numerical model for (2.9) and to understand the intrinsic challenges for an advection-diffusion model is to assume that $v = u\vec{d}$ where $\vec{d}$ is a fixed direction, so the arrangement of the rate of change of the chemical concentration depends directly on a particular arrangement of the population density in space.

This allows us to both think on a possible biological model in which, for example, the organism could produce the chemical itself according to their space arrangement (such that the layout of the physical space plays a role in the growth rate of the chemical itself), and at the same time understand a conservation form set up such in the case of the dissipative Burgers' equation explained in section 6.

We would like to notice that even if the conservative model presented in this thesis is not the same as the tumor angiogenesis model, it allows us to layout the foundations to numerical simulations to conservation laws with the hope to implement similar conservation laws for the tumor angiogenesis model. Moreover, we are hoping that this set up also illuminates space-dependent models, such as curvature dependent chemotaxis models.

CHAPTER 3

GENERALIZED LOGISTIC GROWTH IN 1D

Before moving to the 2D set up, we build numerical methods for a 1D domain and examine the behavior of the system over a simpler domain. Particularly, we look to investigate the effect that α, the exponent in the generalized logistic growth term, $u(1-u^{\alpha})$, has on the rate that the solution reaches a steady state. A steady state is when $\lim_{t\to\infty}\partial_t u = 0$ and represents the solution converging to a final, unchanging, equilibrium. In our 1D chemotaxis model, this represents the endothelial cell density growing to the carrying capacity and the gradient of the TAF approaching 0, i.e., the TAF concentration reaching a constant state. In this chapter we examine (2.10) with a particular set of boundary conditions. Note the change in notation, ∂_x is used to denote a partial derivative with respect to space.

$$\begin{aligned} \partial_x u|_{x\in\partial\Omega} &= 0 \\ v|_{x\in\partial\Omega} &= 0 \end{aligned} \tag{3.1}$$

Above, two different types of boundary conditions are used. The first is a Neumann boundary condition where the directional derivative in the normal direction is specified on the boundary. The second is a Dirichlet boundary condition where the value of the function is prescribed on the boundary. Note that as $v = \frac{c_x}{c} = 0$, this implies that $c_x = 0$ on the boundary, thus these boundary conditions are really Neumann for both the endothelial cells and secreted TAF chemical. We want to note that in [13], it is required that the initial conditions to be $u(t=0) \geqslant 0$ over the entire domain. Moreover, the population density equation satisfies a maximum principle which assures the density for it to be positive at all times. This is biologically relevant as negative population densities are nonphysical.

In the case where the logistic growth of the population is absent and using these same boundary conditions, it is known that the steady state of the solution converges to the initial average of the population density [15]. In [13], it is shown that with the inclusion of the logistic growth to the

population, the system will eventually converge to the carrying capacity of the system. This chapter focuses on the effect of the generalized logistic growth, and in particular, that of the exponent α on the time the solution takes to reach the carrying capacity.

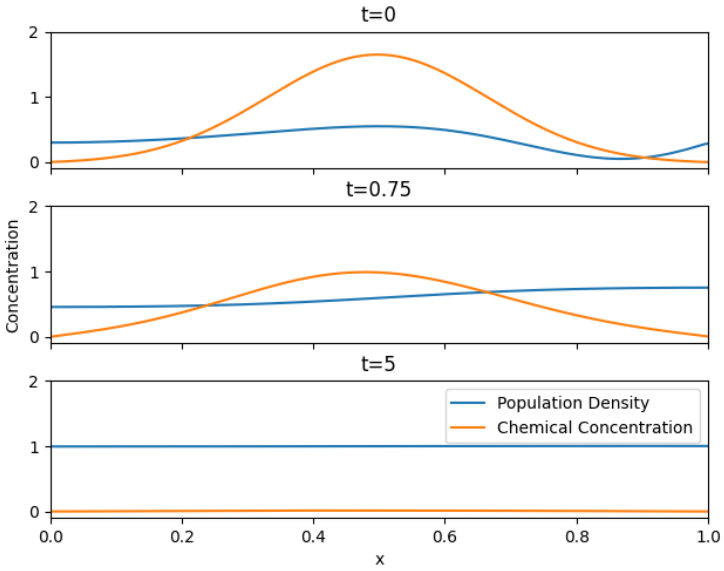

Figure 3.1: Time evolution of chemical and population densities from (2.10) with $\alpha = 1$.

At a certain finite time both u and v flatten out due to the diffusion in the system. Because of this, we conduct a heuristic approach comparing the solution of u in (2.10) with that of the generalized logistic ordinary differential equation (ODE)

$$\partial_t u = ru(1 - u^\alpha). \tag{3.2}$$

Equation (3.2) is a separable ODE and with a general solution

$$u(t) = \frac{C^{\frac{1}{\alpha}} e^{rt}}{(1 + Ce^{\alpha rt})^{\frac{1}{\alpha}}}.$$

Here, $C \in \mathbb{R}$ and r is the non-dimensionalized rate constant from section 2.3. We want to find C such that this general solution describes the growth of a population from some initial density u_0. For $u(0) = u_0$ we get

$$C = \frac{u_0^\alpha}{1 - u_0^\alpha}.$$

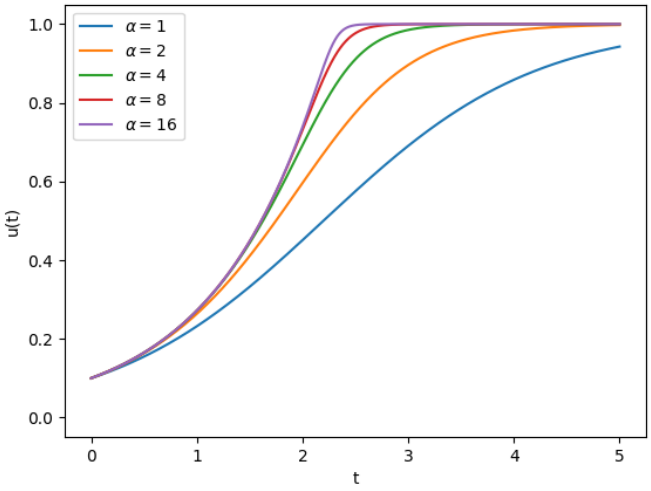

Figure 3.2: (3.3) for different values of α for $t \in [0, 5]$.

Giving the specific solution to this ODE

$$u(t) = \frac{u_0 e^{rt}}{(1 - u_0^\alpha + u_0^\alpha e^{\alpha rt})^{\frac{1}{\alpha}}}. \tag{3.3}$$

Figure 3.2 shows various solutions for different values of α. Notice that varying α changes the rate at which the population reaches the carrying capacity. Also notice that if the initial population is greater than the environmental carrying capacity the population decreases, which is a natural behavior for logistic equations.

3.1 Numerical Simulations

We use an explicit finite difference scheme with second order spatial derivatives and first order temporal derivatives to solve the transformed and non-dimensionalized system numerically in 1D. With Neumann boundary conditions we make sure to utilize 'ghost points' at either end of our interval to facilitate the enforcement of the prescribed derivatives at the boundary.

We solve the equation over the spatial domain $x \in [0, 1]$ and the temporal domain $t \in [0, T]$, where T is a designated stopping time for the system. We choose the spatial and temporal meshes

of $\Delta t = \frac{(\Delta x)^2}{2}$ and $\Delta x < \sqrt{\frac{\varepsilon}{10}}$ respectively. The spatial mesh is chosen to ensure the resolution of all diffusive timescales and the temporal mesh is chosen to be identical to the bound for the heat equation when solved with the same finite difference methods due to the diffusive dominance of this system. Choosing a second order method for the diffusion serves to minimize the effects of numerical diffusivity, which is when the numerical method introduces artificial smoothing at discontinuities or sharp gradients which may not accurately represent the physical diffusion process. By minimizing numerical diffusion, we ensure the actual diffusion of the PDE system is adequately represented. A more in depth explanation of the numerical methods used is available in chapter 4.1 and beyond.

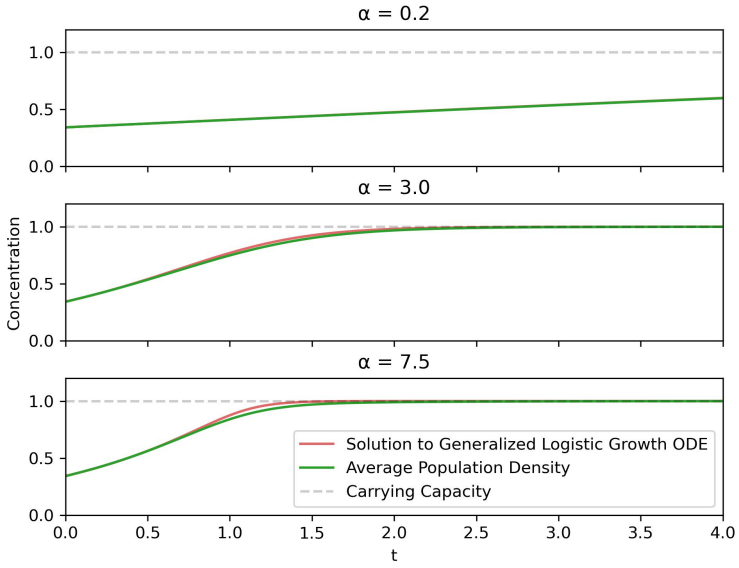

Figure 3.3: Comparing numerical simulations for $\alpha = 0.2, 3.0$, and 7.5 respectively

For small values of α, we find that u quickly diffuses, then it grows similarly to the ODE as it converges to the steady state solution. Conversely, with $\alpha \gtrsim 3$, u grows logistically faster than it can diffuse outwards. For initial conditions that have sufficient variation over the domain, u grows very asymmetrically towards the carrying capacity. See figure 3.4.

A large challenge with this approach is determining at what rate and time u approaches the carrying capacity. One method is taking the 2-norm between u and the carrying capacity within a tolerance ε_{cc}. We also use this method to determine if the solution diffuses to the initial average

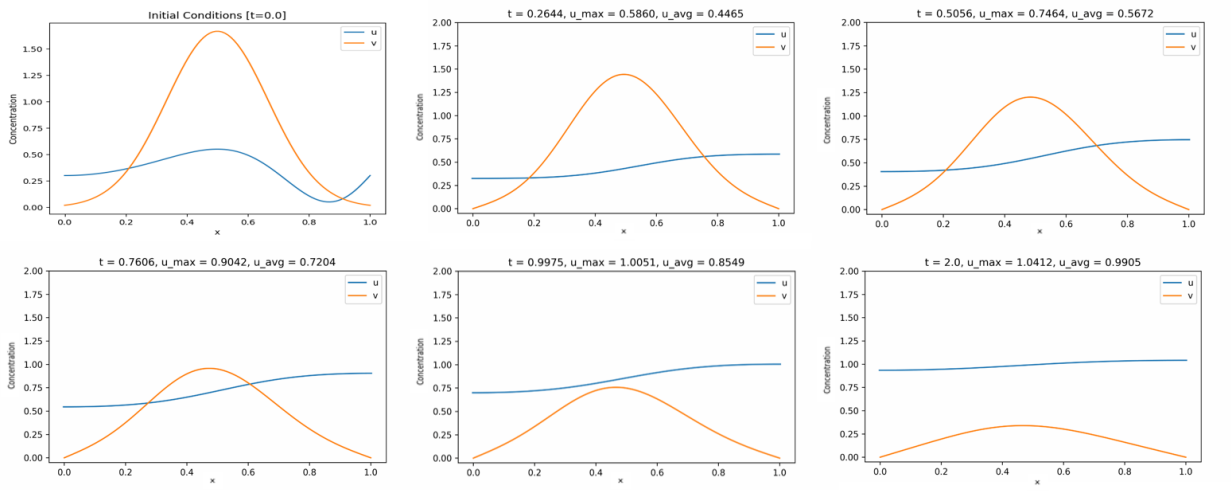

Figure 3.4: $\alpha = 10$, note how u approaches the carrying capacity at different times and never fully diffuses before logistic growth sets in. The x and y axis labels are 'x' and 'Concentration', respectively.

by taking the 2-norm between u and the initial average with a tolerance ε_{avg}. Both norms were evaluated at each timestep throughout the evolution. The first and last time each was reached was recorded. This was especially useful to quantify that time that the solution stayed at the initial average before the logistic growth took over.

3.2 Conclusion and Future Steps

After systematically varying the value of α, we found that the behavior of the system changes drastically when adjusted. The challenges we encountered in quantifying the convergence of the system to the carrying capacity and initial average show the complexity of the system's dynamics and the need for more investigation through numerical and analytical methods.

Future steps in this investigation include extending the analysis of the system and quantifying the current heuristic understanding of how the solutions bifurcate in relation to the α parameter. Also, investigating the system under a larger variety of initial conditions and ratios between the diffusion and growth rates could lead to the discovery of new relationships with the α parameter.

CHAPTER 4

1D ADVECTION-DIFFUSION

The goal of this thesis is to build numerical solvers based on finite difference discretizations for PDE models of chemotaxis. We implement numerical methods for simpler PDE models in 1D and 2D Euclidean space with aims to transfer them to the actual model, equation (2.10), over curved surfaces.

The general expression for the second order linear and quasilinear PDEs we are studying in this thesis is:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial t} + C\frac{\partial^2 u}{\partial t^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial t} + Fu = G. \tag{4.1}$$

Where A, B, C, D, E, F, and G are functions of x and t and A, B, and C are not simultaneously 0. Moreover, we classify various PDEs into three categories based on the relationship between A, B, and C [16]. We define the discriminant, Δ, for the PDE as

$$\Delta = \det \begin{bmatrix} B & 2A \\ 2C & B \end{bmatrix} = B^2 - 4AC. \tag{4.2}$$

If $\Delta > 0$, the PDE is *hyperbolic*; if $\Delta = 0$ the PDE is *parabolic*; and if $\Delta < 0$ the PDE is *elliptic*. The three PDE types: hyperbolic, parabolic, and elliptic have archetypal examples: the wave, heat, and Poisson equations, respectively. We examine the 1D wave equation,

$$\frac{\partial^2 u}{\partial t^2} - \alpha^2 \frac{\partial^2 u}{\partial x^2} = 0,$$

where $\alpha \in \mathbb{R}^+$ and represents the velocity of the wave. We can factor the wave equation:

$$\frac{\partial^2 u}{\partial t^2} - \alpha^2 \frac{\partial^2 u}{\partial x^2} = \left(\frac{\partial u}{\partial t} - \alpha \frac{\partial u}{\partial x}\right)\left(\frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x}\right) = 0, \tag{4.3}$$

resulting in two advection equations. The 1D advection equation,

$$u_t = -\alpha u_x, \tag{4.4}$$

details a packet that moves, or advects, in the $+\hat{x}$ direction at a constant velocity α while preserving the shape of the packet. The direction would be reversed for $\alpha < 0$. The t and x subscripts denote temporal and spatial partial derivatives, respectively. This is a good building block to begin with. From here, it is possible to add modifications to the equation until it represents the desired chemotactic behavior. We begin by adding a diffusive term,

$$u_t = -\alpha u_x + \beta u_{xx}, \tag{4.5}$$

here $\alpha, \beta \in \mathbb{R}^+$. We can see that the advection diffusion equation falls under the definition outlined in equation (4.1) with $A = -\beta$, $D = \alpha$, $E = 1$, and $B, C, F, G = 0$. This PDE is both second order and linear; thus, it is a prime candidate for a finite difference approach.

The solution of this PDE combines both advective and diffusive behaviors. If the initial conditions are a non-constant packet or envelope, this equation will smooth the initial conditions while translating them in the $\hat{x}$ direction.

4.1 Choosing a Finite Difference Scheme

A good strategy for finding the solution to this equation is by finite difference methods. We use finite difference methods because they can provide flexibility with various boundary conditions, analytical stability, and a relatively simple implementation.

To begin creating a finite difference scheme to solve this PDE, it is useful to examine several different finite difference schemes for calculating the spatial and temporal derivatives. Choosing a method has a large impact on stability of results and computational cost. Thus, we carefully consider it.

4.1.1 Introduction to Finite Differences

Finite difference methods are a way for computers to numerically calculate derivatives without having to compute limits. The derivative is defined as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{4.6}$$

Alternatively, finite differences are defined by the Taylor expansion of f at x, displaced by a small distance h:

$$f(x+h) = f(x) + h\frac{f'(x)}{1!} + h^2\frac{f''(\xi)}{2!}. \tag{4.7}$$

Here, $\xi \in [0, h]$ and $\frac{h^2}{2} f''(\xi)$ is an upper bound on the error of the Taylor expansion for f on $[0, h]$.

$$f'(x) = \frac{f(x+h) - f(x)}{h} + h\frac{f''(\xi)}{2!}. \tag{4.8}$$

The error above scales with h. Assuming that $h\frac{f''(\xi)}{2!}$ is small, the first derivative can be approximated

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{4.9}$$

Visually, this is very similar to the definition of the first derivative, which is a good sanity check. The "order" of the approximation comes from the order of h in the error term in the Taylor expansion in (4.8). Because this example has error that scales as h, it is a first order scheme. This is a "forward difference" scheme because h is added to x, and then f is evaluated. We also define the forward, backwards, and centered finite difference operators:

$$\begin{aligned} D_h^+[f] &= \frac{f(x+h) - f(x)}{h}, \\ D_h^-[f] &= \frac{f(x) - f(x-h)}{h}, \\ D_h^C[f] &= \frac{f(x+h) - f(x-h)}{2h}. \end{aligned}$$

If h is subtracted, then a "backwards difference" is obtained and if h is both added and subtracted, then the result is a "central difference".

Central differences are higher order, but they are not suitable for all cases, as seen in the following sections. Higher order schemes are desirable, but as we see in later sections we must also consider the stability and computational cost of the discretization scheme.

To apply finite difference methods to equations such as (4.5), we discretize the spatial domain into n partitions which are equally spaced with a width of h. It is possible to use non-uniform spacing, but that is unnecessary for this application. We discretize the time domain in a similar manner up to N timesteps of length k. Using this spatial discretization, it is possible to use finite

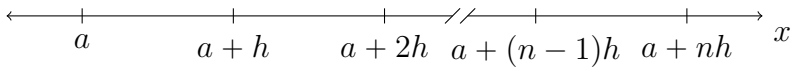

Figure 4.1: Discretizing the spatial domain into n equally spaced partitions of length h.

differences to approximate the value of the derivative at each interval. To do this, we want some way of evaluating this function at each discrete point in the spatial domain.

An introduction of new notation is in order. Δx and Δt now correspond to grid spacing in spatial and temporal discretizations, respectively. We define individual spatial and temporal points on our lattice as $x_j = a + j\Delta x$ and $t_k = t_0 + k\Delta t$. We also introduce a convention for vectors. Bolded characters such as $\mathbf{U}$ represent a vector. When we index with a super-script like $\mathbf{U}^k$ or as $\mathbf{U}(t_k)$, to represent the vector of samples at all the x_j at time t_k. Likewise, we define $\mathbf{U}_j = \mathbf{U}(x_j)$ to be the the vector of samples at the time t_k at all spatial x_j. To denote a specific point, we use the unbolded $U_j^k = U(x_j, t_k)$.

Ideally, we want to be able to represent this equation by applying a matrix, $\mathbb{M}$, to the vector $\mathbf{U}^k$ or by computing a matrix solve with $\mathbb{M}$ and $\mathbf{U}^k$ to time advance the equation to the next timestep. This allows the utilization of efficient algorithms that take advantage of the structure of $\mathbb{M}$ in order to increase the efficiency of the calculations and decrease runtime when compared to other methods

such as using a 'for loop' to iterate through both space and time.

Time Advancement and Implicit vs. Explicit Methods

All of the following methods mentioned involve the discretization of the temporal domain in much of the same way as the spatial domain. For the entirety of this thesis, we will use a first order explicit method to time advance, because it simplifies the analysis by avoiding having to solve a system of equations at each timestep.

So far, we have discussed spatial discretization but we have not mentioned how to evolve the equation forward in time. The easiest method of evolving an equation is to solve the equation in space at a point or points in time, and then use those known values to calculate future values. This is called an *explicit* method because all the values are explicitly known before the next values are calculated. We can define this timestep advancement of U_j^k in terms of a function based on values of $\mathbf{U}^k$. One possible example is: $U_j^{k+1} = f(U_{j-1}^k, U_j^k, U_{j+1}^k)$, but the method could use points from a larger portion of the domain or even previous timesteps.

The another option is an *implicit* method, where the values are not known and instead found by solving a system of equations. Here, we can express the timestep advancement of U_j^k in terms of a function based on values of $\mathbf{U}^k$ and $\mathbf{U}^{k+1}$: $U_j^k = g(U_{j-1}^k, U_j^k, U_{j+1}^k, U_j^{k+1})$, where we would then algebraically or iteratively solve for U_j^{k+1}. Again, g is just one example of a possible scheme. Implicit methods are generally more computationally expensive but they can provide additional stability for stiff ODEs and most evolution PDEs. This allows us to take much larger timesteps than with comparable explicit methods [17]. Because of the computational cost and stability trade-offs between explicit and implicit methods, there exists a final category of *Implicit-Explicit* or *Semi-Implicit* methods which apply a mix of both approaches to reach a desirable compromise. We utilize an explicit first order Forward Euler scheme to move forward in time. With our new notation:

$$\frac{\partial u}{\partial t} \approx \frac{U^{k+1} - U^k}{\Delta t}. \tag{4.10}$$

Notice that this equation is a direct application of (4.9) but over time, not space. We can substitute

this into equation (4.5) and approximate the future timestep, U^{k+1}:

$$U^{k+1} \approx U^k + \Delta t(-\alpha u_x + \beta u_{xx}). \tag{4.11}$$

Now, all that is required are appropriate finite difference methods to represent u_x and u_{xx}.

4.1.2 Finite Difference Methods for Hyperbolic and Parabolic PDEs

Upwind Methods

Upwind methods are a stable (in one direction) first order method of evolving the advection equation by using a finite difference method. The finite difference implementation then takes the form:

$$\begin{aligned} \frac{U_j^{n+1} - U_j^n}{\Delta t} &= -\alpha \frac{U_j^n - U_{j-1}^n}{\Delta x} \\ U_j^{n+1} &= U_j^n - \alpha \frac{\Delta t}{\Delta x} \left(U_j^n - U_{j-1}^n \right). \end{aligned} \tag{4.12}$$

The domain of dependence is a right triangle which can flip direction based on the sign of α, which describes the direction of the fluid flow. For a positive α, the scheme produces a tree as in figure 4.2.

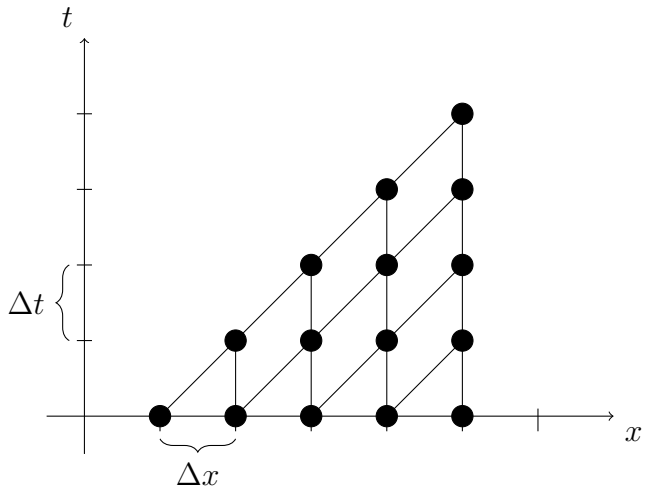

Figure 4.2: Domain of dependence for the upwind method.

Domain of Dependence

The domain of dependence refers to the region of the domain from which information is used to update a solution at a particular point. Note that this is slightly different than the standard definition than when dealing with analytical PDEs. Looking at a small section of a domain, we pick a point and slowly move our system forward in time by single timestep increments. At each increment, we trace the dependency of the point(s) in question based on equation (4.12), which produces a tree-like structure that branches down in figure 4.2. Examining this structure can provide insights into the stability of the chosen method. If we attempt to represent the spatial derivative with a centered difference implementation:

$$f'(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}. \tag{4.13}$$

Equation (4.13) appears reasonable, but an issue arises.

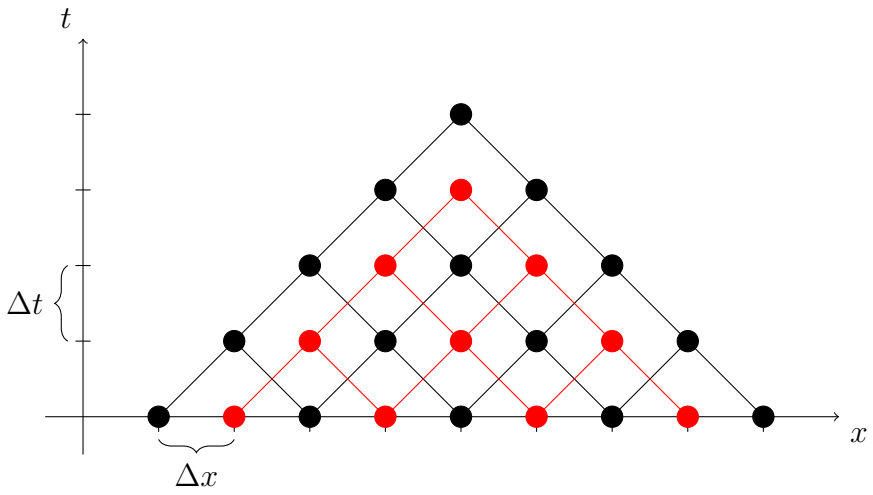

Figure 4.3: Domain of dependence for a first order central difference scheme for the advection equation. Note that the slopes of the black and red lines are $\pm\alpha$.

Figure 4.3 highlights an issue: there are two disconnected domains in the region of interest. Moreover, no information is passed between neighboring partitions which leads to instability.

Stability of Finite Difference Methods

We have introduced the discrete domain of dependence with the figures above, but there is a continuous analogue that is important to discuss. We imagine a packet transported by the advection equation. It does not diffuse and thus remains in a constant shape. On a position versus time plot, we can pick a point on the packet and trace out the straight line that it travels through space and time. This is a characteristic line. We also know that because there is no diffusion in the equation, the value of the point does not change throughout time, so this characteristic remains constant.

When we solve this system numerically, we introduce both approximation errors in our finite differences and roundoff errors when the computer system operates on the numbers. We expect, and show in this chapter, that as we make a finer mesh in space our finite differences get more accurate; but it is not obvious that these errors do not excessively compound and ruin the simulation.

This is further discussed in section 4.4, but we desire that for a small perturbation in the initial conditions, ε, from the true solution that after a time, t, the magnitude of the difference between the two solutions remains bounded by $L\varepsilon$, where $L \in \mathbb{R}^+$. This ensures that if a small amount of error is introduced, i.e., roundoff error, the solution remains close to the true solution.

We can combine this notion with the characteristic defined above. In figure 4.3, we examine the black and red points separated by Δx. If we take Δx to be small enough, we should be able to assume that the solution at the black and red characteristics we trace out should be remain close. However, because the black and red structures are uncoupled, this is not necessarily the case which leads to instability. Alternatively, we examine figure 4.2 and see that there is only one structure, and that when the characteristics get arbitrarily close they remain close. The only remaining step is to put a bound on Δt that ensures stability.

Bounding Timesteps and the CFL Condition

We also take steps to determine the bounds for Δt using the domain of dependence. For hyperbolic behavior, we want any advective behavior to remain inside the domain of dependence, otherwise information would be lost. Let the green triangle in figure 4.4 represent a packet of information

that is advecting rightwards. The blue and red arrows are two possible speeds the packet could be moving, at speeds α_1 for blue and α_2 for red. The blue and red arrows are also examples of possible characteristic lines the packet follows. If the packet moves at speed α_2, it escapes the domain of dependence and becomes unresolved. This can be seen because it moves more spatial steps than timesteps, thus inevitably skipping over certain spatial steps. However, if the packet moves at speed α_1 or less, it will always remain inside the domain of dependence until it moves past the point in question in the spatial domain. Through geometry, it can be seen this limit is $\alpha_1 \leqslant \frac{\Delta x}{\Delta t}$, which can then be rearranged in terms of Δt as

$$\Delta t \leqslant \frac{\Delta x}{\alpha}. \tag{4.14}$$

This is known as the Courant-Friedrichs-Lewy Condition, or CFL Condition. When we combine

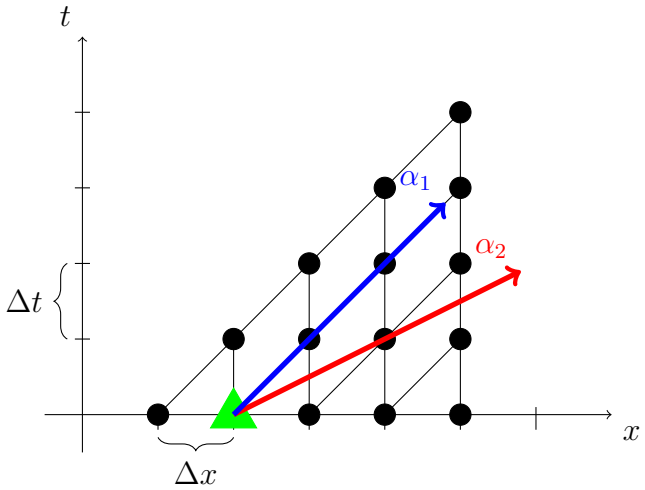

Figure 4.4: Domain of dependence for the upwind method with travelling information packet.

our upwind scheme for the hyperbolic portion with the parabolic scheme we derive in the next section, we will need to consider the CFL condition and the analogous Δt restriction we find for the parabolic scheme in 4.4 to make a heuristic bound on Δt for the combined advection diffusion equation.

Central Difference Formulation for Parabolic Equations

Next, we consider parabolic behavior and how to represent it with finite differences. In the beginning of this chapter, we said that the archetypal parabolic problem was the heat, or diffusion equation,

$$\frac{\partial u}{\partial t} = \beta \frac{\partial^2 u}{\partial x^2}. \tag{4.15}$$

Equation (4.15) contains a second derivative which we have not yet found by finite difference methods. Unlike the advection equation where we expect the solution to travel in only one direction at at a time, the diffusion equation smears the solution in both directions along the spatial domain. Thus, we want a central difference method so that both sides are taken into account. We start by Taylor expanding both $f(x+h)$ and $f(x-h)$ up to the fourth order.

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2 f''(x)}{2} + \frac{h^3 f'''(x)}{6} + \frac{h^4 f^{(4)}(\xi_+)}{24} \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2 f''(x)}{2} - \frac{h^3 f'''(x)}{6} + \frac{h^4 f^{(4)}(\xi_-)}{24} \end{aligned}$$

ξ_+ and ξ_- are values on the domains $[0, h]$ and $[-h, 0]$ respectively. We define $\xi \in [-h, h]$. Adding the above two equations cancels the first and third order terms and then isolating the second derivative:

$$\begin{aligned} f(x+h) + f(x-h) &= 2f(x) + h^2 f''(x) + \frac{2h^4 f^{(4)}(\xi)}{24}, \\ f''(x) &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \frac{h^2 f^{(4)}(\xi)}{12}, \\ f''(x) &\approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \end{aligned}$$

Thus, we have found a central second order finite difference method for the second derivative. We also take this opportunity to define another finite difference operator:

$$D_{h^2}^{2,C}[f] = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

We possess all the requisite pieces of the advection diffusion equation in finite difference form, now we must assemble them in a way that is computationally efficient.

4.2 Solving with Linear Algebra Methods

4.2.1 Developing the Time Advancement Matrix

We want to solve equation (4.5) with methods from linear algebra because there exist faster algorithms for computing the solution. In 1D, this is only marginally faster than a 'for loop', but in two and higher dimensions, these methods greatly decrease computation time. Our goal is to express each timestep as a matrix, $\mathbb{M}$, operating on our solution vector, $\mathbf{U}^k$:

$$\mathbf{U}^{k+1} = \mathbb{M}\mathbf{U}^k. \tag{4.16}$$

Throughout this thesis, matrices are represented as upper case letters with the double bar on the leading edge with the exception of the identity matrix which is represented by I. We can express our initial PDE, equation (4.5), with the finite differences that we found in the previous section. Using our first order upwind for advection and second order centered difference for diffusion as well as Forward Euler for the time derivative we get:

$$\frac{U_j^{k+1} - U_j^k}{\Delta t} = -\alpha \left(\frac{U_j^k - U_{j-1}^k}{\Delta x} \right) + \beta \left(\frac{U_{j+1}^k - 2U_j^k + U_{j-1}^k}{\Delta x^2} \right). \tag{4.17}$$

We solve for the next timestep:

$$U_j^{k+1} = U_j^k + \Delta t \left[-\alpha \left(\frac{U_j^k - U_{j-1}^k}{\Delta x} \right) + \beta \left(\frac{U_{j+1}^k - 2U_j^k + U_{j-1}^k}{\Delta x^2} \right) \right]. \tag{4.18}$$

Define square $N \times N$ matrices $\mathbb{A}$ and $\mathbb{D}$, representing advection and diffusion respectively, as

$$\mathbb{A} = \frac{\alpha}{\Delta x}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots \\ -1 & 1 & 0 & 0 & 0 & \cdots \\ 0 & -1 & 1 & 0 & 0 & \cdots \\ 0 & 0 & -1 & 1 & 0 & \ddots \\ 0 & 0 & 0 & -1 & 1 & \ddots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}, \; \mathbb{D} = \frac{\beta}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots \\ 1 & -2 & 1 & 0 & 0 & \cdots \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ 0 & 0 & 1 & -2 & 1 & \ddots \\ 0 & 0 & 0 & 1 & -2 & \ddots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.19}$$

The structure of $\mathbb{A}$ and $\mathbb{D}$ comes directly from the indexing on the calculated finite differences. Each row corresponds to a single entry in $\mathbf{U}^k$. Traversing down the rows shows how the pattern exhibited by the indices in equation (4.18), is applied to each entry in $\mathbf{U}^k$. The diagonal structures allows the products $\mathbb{A}\mathbf{U}^k$ and $\mathbb{D}\mathbf{U}^k$ to take into account only the neighboring values of each index. The equation can be rewritten with the vector $\mathbf{U}^k$ instead of indices as:

$$\begin{aligned} \mathbf{U}^{k+1} &= \mathbf{U}^k + \Delta t\left[-\mathbb{A}\mathbf{U}^k + \mathbb{D}\mathbf{U}^k\right] \\ \mathbf{U}^{k+1} &= \left[I + \Delta t\left(-\mathbb{A} + \mathbb{D}\right)\right]\mathbf{U}^k \\ \mathbf{U}^{k+1} &= \mathbb{M}\mathbf{U}^k \end{aligned} \tag{4.20}$$

Here, $\mathbb{M}$ is a tridiagonal matrix. At each step, we apply $\mathbb{M}$ to the solution vector $\mathbf{U}^k$. The matrix is of the following form, where m_{-1}, m_0, and m_1 are non-zero values determined by the constants $\beta, \alpha, \Delta x$, and Δt that are constant along diagonal and sub/super diagonals.

$$m_{-1} = \Delta t\left(\frac{\alpha}{\Delta x} + \frac{\beta}{\Delta x^2}\right) \quad m_0 = 1 + \Delta t\left(\frac{-\alpha}{\Delta x} - 2\frac{\beta}{\Delta x^2}\right), \quad m_1 = \Delta t\left(\frac{\beta}{\Delta x^2}\right) \tag{4.21}$$

$$\mathbb{M} = \begin{bmatrix} m_0 & m_1 & 0 & 0 & \cdots & 0 \\ m_{-1} & m_0 & m_1 & 0 & \cdots & 0 \\ 0 & m_{-1} & m_0 & m_1 & & 0 \\ 0 & 0 & m_{-1} & m_0 & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & m_1 \\ 0 & 0 & 0 & 0 & m_{-1} & m_0 \end{bmatrix}$$

4.3 Implementing Boundary Conditions

Until now, we have neglected boundary conditions. For simplicity, let us continue to constrict our spatial domain to $[0, 1]$. Also define $\mathbf{B}$ as a time-independent vector with the same length as $\mathbf{U}^k$. With certain changes to the chosen scheme, $\mathbf{B}$ could be time-dependent but that is outside the scope of this thesis. To implement the boundary value vector, we want to simply add it to the right hand side. The boundary value vector is multiplied by a factor of Δt because it is part of time advancement.

$$\mathbf{U}^{k+1} = \mathbb{M}\mathbf{U}^k + \Delta t\mathbf{B} \tag{4.22}$$

We define l, l_x, r, r_x as the fixed values of $\mathbf{U}^k$ or the fixed derivative of $\mathbf{U}^k$ on the boundaries. The three types of boundary conditions we will examine in this thesis are:

1. Dirichlet: $U(x = 0) = l$, $U(x = 1) = r$
2. Neumann: $\partial_x U(x = 0) = l_x$, $\partial_x U(x = 1) = r_x$
3. Periodic: $U(x = 0) = U(x = 1)$, $\partial_x U(x = 0) = \partial_x U(x = 1)$

As detailed in sections 4.3 and 4.3, by altering the size of $\mathbf{U}^k$ and $\mathbb{M}$ by adding or removing points for Dirichlet or Neumann problems, we consider what the finite difference implementation should be on the boundaries, and the values that are not found in $\mathbb{M}$ are resolved by the addition of $\mathbf{B}$.

Dirichlet Boundaries

Beginning with Dirichlet conditions, we are looking for the solution to obtain specific boundary values on the domain $[0, 1]$ where l and r are the time-independent left and right boundary values. For the boundary conditions:

$$U(x = 0) = l,\ U(x = 1) = r$$

Out of N points, only $N - 2$ are unknown because the boundaries are known at any given time. Thus, we apply a size $N-2 \times N-2$ matrix to our solution to obtain the following timestep. Given the previous $\mathbb{M}$ matrix where m_{-1}, m_0, and m_1 are the diagonal elements, we then add wanted boundary conditions we want after as a vector, $\mathbf{B}_D = \langle \mathbf{B}_0, 0, ..., 0, \mathbf{B}_n \rangle$. If the boundaries are set to 0, then $l = 0 = r$ then $\mathbf{B}$ is the 0-vector.

On the left boundary at $x = 0$, the finite difference scheme for U_j^k includes values for both advective and diffusive behavior from the left boundary due to the direction of advection moving left-to-right. Conversely, $U_{(n-1)j}^k$ only contains information concern diffusion because information on the boundary will not advect to the left due to $\alpha > 0$.

$$\mathbb{M}_{(N-2\times N-2)} = \begin{bmatrix} m_0 & m_1 & 0 & 0 & \cdots & 0 \\ m_{-1} & m_0 & m_1 & 0 & \cdots & 0 \\ 0 & m_{-1} & m_0 & m_1 & \cdots & 0 \\ 0 & 0 & m_{-1} & m_0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & m_1 \\ 0 & 0 & 0 & 0 & m_{-1} & m_0 \end{bmatrix},\ \mathbf{B}_D = \begin{bmatrix} \left(\frac{\beta}{\Delta x^2} + \frac{\alpha}{\Delta x}\right) l \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \frac{\beta}{\Delta x^2} r \end{bmatrix}$$

$$\mathbf{U}^{k+1} = \mathbb{M}_{(N-2\times N-2)} \mathbf{U}^k + \Delta t \mathbf{B}_D$$

Neumann Boundaries

For Neumann boundary conditions, we want the first derivative to have specific values on the boundaries for the domain $[0, 1]$, l_x and r_x.

$$\partial_x U(x = 0) = l_x,\ \partial_x U(x = 1) = r_x$$

Similarly to the Dirichlet scheme, we want a boundary vector $\mathbf{B}_N = \langle \mathbf{B}_{x,0}, 0, ..., 0, \mathbf{B}_{x,n} \rangle$.

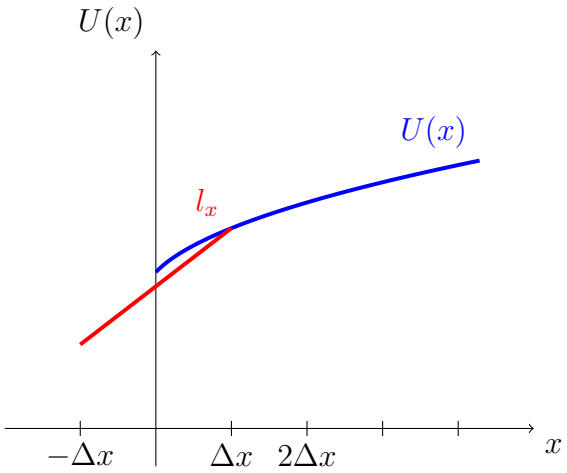

Figure 4.5: Left hand boundary for Neumann boundary conditions. Blue: $U(x)$, red: extended l_x.

In order to calculate the values of $\mathbf{B}_N$, we use a centered difference method. We want to know $\mathbf{U}^k$ on $[0, 1]$, and using a centered difference for the derivative on the boundaries requires the addition of the ghost points, extending the domain by Δx on each side from $[0, 1]$ to $[-\Delta x, 1 + \Delta x]$. In order to evaluate the boundary at $x = 0$ we need to find U^k_{-j}. Because the values of the derivative of $\mathbf{U}^k$ at the boundary and U^k_j are known it is simple to calculate U^k_{-j} by using the equation of a line in point-slope form.

$$\begin{aligned} U^k_j - U^k_{-j} &= l_x(\Delta x - (-\Delta x)) \\ U^k_{-j} &= U^k_j - 2l_x\Delta x \end{aligned} \tag{4.23}$$

By the same process, the right boundary results in

$$U^k_{(n+1)j} = U^k_{(n-1)j} + 2r_x\Delta x \tag{4.24}$$

Above, U^k_j and $U^k_{(n-1)j)}$ are not constants, they are part of $\mathbf{U}^k$, so they are represented in $\mathbb{M}_{(N+2\times N+2)}$ where certain values are multiplied by two in the first and last rows. Conversely, the values $-2l_x\Delta x$ and $2r_x\Delta x$ are constants that cannot be included in the matrix and are thus the components of the boundary vector.

$$\mathbb{M}_{(N+2\times N+2)} = \begin{bmatrix} m_0 & 2*m_1 & 0 & 0 & \cdots & 0 \\ m_{-1} & m_0 & m_1 & 0 & \cdots & 0 \\ 0 & m_{-1} & m_0 & m_1 & & 0 \\ 0 & 0 & m_{-1} & m_0 & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & m_1 \\ 0 & 0 & 0 & 0 & 2*m_{-1} & m_0 \end{bmatrix}, \ \mathbf{B}_N = \begin{bmatrix} -2l_x\Delta x \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 2r_x\Delta x \end{bmatrix}$$

$$\mathbf{U}^{k+1} = \mathbb{M}_{(N+2\times N+2)}\mathbf{U}^k + \Delta t\mathbf{B}_N$$

However, an issue arises when solving the advection diffusion equation using Neumann boundary conditions. The PDE does not have a unique solution. Let $w(x,t)$ be a solution to equation (4.5) with Neumann boundary conditions. Then, $w(x,t) + C$, where $C \in \mathbb{R}$ is also a solution to (4.5). This can be seen by substituting $w(x,t)$ and $w(x,t) + C$ into (4.5) and noticing that C falls out. Dirichlet boundary conditions do not encounter this problem because specifying the boundary fixes $C = 0$ forcing the solution to be unique.

One option to fix this issue is adjusting the boundary conditions from Neumann to Robin, which combines Neumann and Dirichlet characteristics, i.e., $U(x = 0) + \partial_x U(x = 0) = l$ and $U(x = 1) + \partial_x U(x = 1) = r$. This fixes the boundary and forces a unique solution. For problems

modeling different physical phenomena, one may also put a constraint on the interior of the domain which would also fix this issue.

Periodic Boundaries

For periodic boundary conditions, the solution on the boundaries should be equal.

$$U(x=0) = U(x=1)$$

With a size $(N \times N)$ system, we adapt our $\mathbb{M}$ matrix as such with values in the corners such that $U(x=0)$ and $U(x=1)$ both wrap around to the other side to create a cyclic matrix. This allows for the $\mathbf{U}^k$ vector to wrap around the domain and pass information.

$$\mathbb{M} = \begin{bmatrix} m_0 & m_1 & 0 & 0 & \cdots & m_{-1} \\ m_{-1} & m_0 & m_1 & 0 & \cdots & 0 \\ 0 & m_{-1} & m_0 & m_1 & & 0 \\ 0 & 0 & m_{-1} & m_0 & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & m_1 \\ m_1 & 0 & 0 & 0 & m_{-1} & m_0 \end{bmatrix}$$

$$\mathbf{U}^{k+1} = \mathbb{M}\mathbf{U}^k$$

4.4 Checking Validity of Numerical Solutions

Picking an appropriate Δt is important. Too large of a Δt with an explicit scheme leads to exponential increases in errors as seen in the next section; too small of a Δt is computationally inefficient. Thus, it is necessary we choose a Δt that is appropriately sized to ensure accuracy in the solution and not waste computing time with too short a timestep.

Von Neumann Analysis

Von Neumann Stability Analysis (vNSA) is a tool to examine the stability of linear finite difference schemes. Because the schemes we are examining are linear, linear combinations of solutions are solutions as well. Just as how we can use Fourier Series to solve PDEs, we can use a Discrete Fourier Series to evaluate the stability of these discrete finite difference operators that we derived. All we must do is expand the errors produced by these discrete operators, and then plug them back into the operator. In order for a scheme to be convergent, we expect that its error will decrease over time as it converges. Thus, we have a simply criterion for convergence using vNSA. Let $\varepsilon_x^k = E(t)e^{iKj}$ be the error of the chosen finite difference method at time k and spatial point j. Here we define the wavenumber to be K as we index in time by k. Then for the method to be convergent it is necessary:

$$\begin{aligned} &\left|\frac{\varepsilon_j^{k+1}}{\varepsilon_j^k}\right| \leqslant 1 \\ \implies &\left|\frac{E(k+1)}{E(k)}\right| \leqslant 1 \end{aligned} \tag{4.25}$$

Note that if the inequality in equation (4.25) was reversed such that $\left|\frac{E(k+1)}{E(k)}\right| \geqslant 1$, $E(k)$ would grow unbounded and exponentially. We can apply vNSA to the second order central difference formulation we found for the diffusion operator.

$$\varepsilon_j^{k+1} = \varepsilon_j^k + \frac{\beta\Delta t}{\Delta x^2}(\varepsilon_{j+1}^k - 2\varepsilon_j^k + \varepsilon_{j-1}^k) \tag{4.26}$$

Using the definition of ε_x^k from above:

$$\varepsilon_j^{k+1} = E(k+1)e^{iKx}, \quad \varepsilon_j^k = E(k)e^{iKx}, \quad \varepsilon_{j+1}^k = E(k)e^{iK(x+\Delta x)}, \quad \varepsilon_{j-1}^k = E(k)e^{iK(x-\Delta x)} \tag{4.27}$$

Substituting them into (4.26)

$$E(k+1)e^{iKx} = E(k)e^{iKx} + \frac{\beta\Delta t}{\Delta x^2}(E(k)e^{iK(x+\Delta x)} - 2E(k)e^{iKx} + E(k)e^{iK(x-\Delta x)}) \tag{4.28}$$

Now we solve for $\frac{E(k+1)}{E(k)}$ using Euler's Identity:

$$\frac{E(k+1)}{E(k)} = 1 + \frac{\beta\Delta t}{\Delta x^2}(e^{iK\Delta x} - 2 + e^{-iK\Delta x})$$
$$\frac{E(k+1)}{E(k)} = 1 + \frac{\beta\Delta t}{\Delta x^2}\left[-4\sin^2\left(\frac{K\Delta x}{2}\right)\right]$$

We need $\left|\frac{E(k+1)}{E(k)}\right| \leqslant 1$ and we know $\sin^2$ is bounded between 0 and 1.

$$\left|-4\frac{\beta\Delta t}{\Delta x^2}\left[\sin^2\left(\frac{K\Delta x}{2}\right)\right]\right| \leqslant 2$$
$$\frac{\beta\Delta t}{\Delta x^2}\left[\sin^2\left(\frac{K\Delta x}{2}\right)\right] \leqslant \frac{1}{2}$$

$$\Delta t \leqslant \frac{\Delta x^2}{2\beta} \tag{4.29}$$

Thus, if we only implemented our second order central difference scheme for diffusion, in order to guarantee the stability the timestep, Δt, would need to be less than or equal to $\frac{\Delta x^2}{2\beta}$. Anything larger would be unstable.

Comparison to Elliptic Solution

The final class of PDEs that we examined at the beginning of this chapter are elliptic PDEs. The simplest elliptic problem is Laplace's equation. In 1D it is written as

$$\frac{\partial^2 u}{\partial x^2} = 0. \tag{4.30}$$

Elliptic PDEs describe steady state behaviors and generally do not have time derivatives. Because of this, they are most frequently applied to steady state phenomena like electrostatics, steady state heat conduction, or structural mechanics. Although our advection diffusion equation is an evolution equation, we expect it to reach a steady state in the limit of large t. There are a few reasons for this. First, our boundary conditions are constant; second, we have no time-dependent nonhomogeneous addition to our advection diffusion equation; and third, we expect our final PDE model of

chemotaxis to approach a steady state.

We want our advection diffusion equation to reach a steady state solution because then the time derivative would vanish and reduce the dimension of the PDE to an ODE which is analytically solvable. We can then solve a boundary value ODE with the matrices and boundary conditions that we developed in sections 4.3. Because we can obtain an analytical solution to the ODE problem, we can compare our numerical and analytical solution to establish if our numerical methods worked, and to what accuracy they performed.

As $t \to \infty$, the solution to the advection diffusion equation approaches a steady-state elliptic solution:

$$u_t = 0 = -\alpha u_x + \beta u_{xx}. \tag{4.31}$$

As a result, it can be treated as an ODE and which has a known general solution,

$$u(x) = C_1 e^{\frac{\alpha}{\beta}x} + C_2, \tag{4.32}$$

with $C_1, C_2 \in \mathbb{R}$. From the general solution, boundary conditions can be inputted and a solution to the steady state problem found. For Dirichlet boundary conditions,

$$u(0) = l, \quad u(1) = r. \tag{4.33}$$

Here, l and r are the same time-independent constants as above. We can use these boundary conditions to solve for the constants C_1 and C_2 in (4.32).

$$l = C_1 e^{\frac{\alpha}{\beta}(0)} + C_2, \quad r = C_1 e^{\frac{\alpha}{\beta}(1)} + C_2$$

$$u(x) = \left(l - \frac{r - l e^{\frac{\alpha}{\beta}}}{1 - e^{\frac{\alpha}{\beta}}} \right) e^{\frac{\alpha}{\beta}x} + \frac{r - l e^{\frac{\alpha}{\beta}}}{1 - e^{\frac{\alpha}{\beta}}} \tag{4.34}$$

Now, we need a way to check that the long time behavior of our numerical solution to (4.5) to ensure it matches the analytical one. Because we are in the limit of $u_t = 0$, we want to strip our

system detailed in section 4.3 of any mention of time. We also add in functionality to solve the non-homogeneous equation. The non-homogeneous portion of the equation is detailed below as $f(x)$. We choose to implement a non-homogeneous term in the equation because it has physical relevance. In PDE equations, a constant non-homogeneous term often represents a 'source' in the system, i.e., a heat source for a heat equation, a driving force in the wave equation, or in the chemotactic model an internal change in cellular population density by logistic growth.

We begin to discretize the non-homogeneous advection diffusion equation by the same methods we used earlier in the chapter. We want to use the same finite difference methods as in the parabolic case.

$$u_t = 0 = -\alpha u_x + \beta u_{xx} + f(x)$$

or

$$\alpha u_x - \beta u_{xx} - f(x) = 0 \tag{4.35}$$

This translates to the finite difference scheme

$$\alpha \left(\frac{U_j^t - U_{j-1}^t}{\Delta x} \right) - \beta \left(\frac{U_{j+1}^t - 2U_j^t + U_{j-1}^t}{\Delta x^2} \right) - f(x_j) = 0 \tag{4.36}$$

Now, we have a new tridiagonal matrix, $\mathbb{M}_{\mathrm{E}}$, 'E' for elliptic, with subdiagonals $m_{E,-1}$, $m_{E,0}$, and $m_{E,1}$.

$$m_{E,-1} = \frac{-\alpha}{\Delta x} - \frac{\beta}{\Delta x^2} \quad m_{E,0} = \frac{\alpha}{\Delta x} + 2\frac{\beta}{\Delta x^2}, \quad m_{E,1} = \frac{-\beta}{\Delta x^2} \tag{4.37}$$

The subdiagonal structure closely resembles the ones from the non-elliptic problem above.

$$\mathbb{M}_{\mathrm{E}} = \begin{bmatrix} m_{E,0} & m_{E,1} & 0 & 0 & \cdots & 0 \\ m_{E,-1} & m_{E,0} & m_{E,1} & 0 & \cdots & 0 \\ 0 & m_{E,-1} & m_{E,0} & m_{E,1} & & 0 \\ 0 & 0 & m_{E,-1} & m_{E,0} & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & m_{E,1} \\ 0 & 0 & 0 & 0 & m_{E,-1} & m_{E,0} \end{bmatrix} \tag{4.38}$$

We also define two vectors of length N: a boundary vector, $\mathbf{B} = \langle \mathbf{B}_0, 0, ..., 0, \mathbf{B}_n \rangle$, and a solution vector, $\mathbf{S}$, where the jth element of S is the non-homogeneous portion, $f(x_j)$. We hope to solve the equation

$$\mathbb{M}_E \mathbf{U}^k + \mathbf{B} = \mathbf{S} \tag{4.39}$$

for $\mathbf{U}^k$. After producing $\mathbf{U}^k$, we compare the numerical solution to the analytical solution, such as (4.34) in the homogeneous case, setting $S_i = 0$, or derive a new analytical solution for a non-homogeneous problem. The same steps can be taken with Robin boundary conditions to correct the non-uniqueness of Neumann boundary conditions as above.

In figure 4.6 we compute the log of the absolute error between the numerical and analytical solution to the elliptic problem. Notice that as n, the number of spatial discretizations, increases the accuracy of the numerical solution does as well, indicative of convergence. Additionally, the error decreases proportional to $h = 1/n$ meaning that we have first order convergence. This is expected because the upwind method that we use is first order. It is also the lowest order method that we use because our second derivative discretization is second order. Thus, the error converges as we expect. This is a good check to see that our finite difference methods are working correctly and we can move on to solving the parabolic equation.

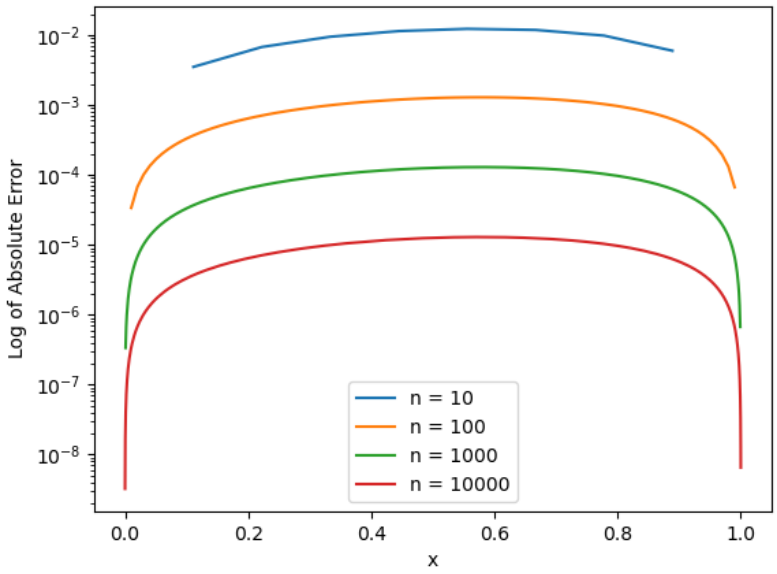

Figure 4.6: Error for the homogeneous elliptic problem with different spatial discretizations

4.5 Numerical Solution to the 1D Non-homogeneous Advection Diffusion Equation with Dirichlet Boundary Conditions

We proceed to solve the 1D nonhomogeneous advection diffusion equation with Dirichlet boundary conditions,

$$u_t + \alpha u_x - \beta u_{xx} - f(x) = 0. \tag{4.40}$$

We again choose to add the nonhomogeneous term for greater applicability. We also choose to only include results for the Dirichlet problem with $l = 0.25$ and $r = 0.75$ for brevity. The methods are much the same for the Robin or periodic case and can be implemented from the matrices described earlier in this chapter.

Figure 4.7 shows snapshots of numerical solutions to a choice problem, $u_t + u_x - \frac{1}{2}u_{xx} = \frac{1}{2}x$, with Dirichlet boundary conditions. We see that as time progresses, the solution approaches a steady state equilibrium as expected and also converges to an analytical solution as desired.

Overall, we made great progress in building a finite difference framework for chemotaxis in this chapter. We developed a definition and some examples of finite difference methods, looked at the

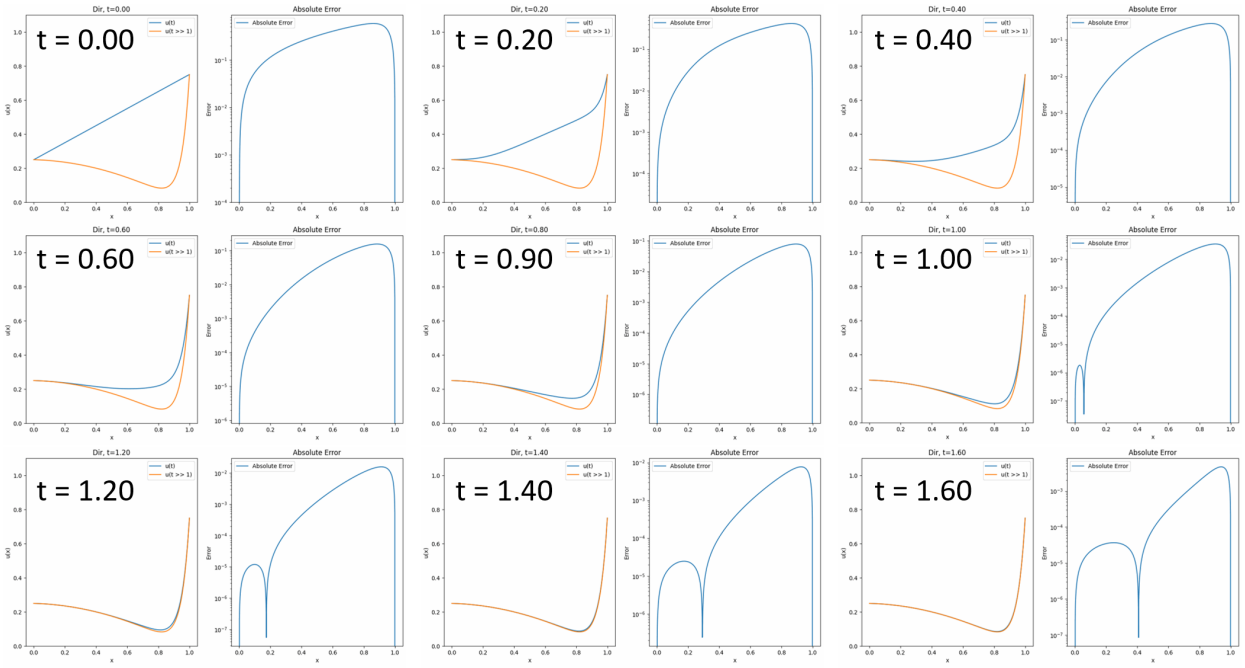

Figure 4.7: Nine snapshots of the parabolic Dirichlet problem for the non-homogeneous advection diffusion equation $u_t + u_x - \frac{1}{2}u_{xx} = \frac{1}{2}x$. The left plot is the numerical solution in blue and the analytical steady-state in orange. The right plot is the absolute error between the numerical and analytical solutions. Note that the right plots have different scaling on the y axes.

convergence criterion for the various methods as well as pitfalls to avoid, and we implemented them successfully to numerically solve a non-homogeneous advection diffusion equation with Dirichlet boundary conditions. In the following chapter, we extend these methods to the 2D advection diffusion equation and begin working in flat Euclidean space.

CHAPTER 5

2D ADVECTION-DIFFUSION

We now extend our 1D results to 2D. We continue to use the advection-diffusion equation, but now we solve it in a domain comprised of three dimensions: x, y, and t.

$$\begin{aligned} u_t &= -\vec{\alpha} \cdot \nabla u + \beta \nabla^2 u \\ u_t &= -\vec{\alpha} \cdot \langle u_x, u_y \rangle + \beta(u_{xx} + u_{yy}) \end{aligned} \tag{5.1}$$

Here, α is now a vector with two positive entries $\vec{\alpha} = \langle \alpha_x, \alpha_y \rangle$. Dotting α into ∇u is the same as two advection equations: one in the $\hat{x}$ direction, and the other in the $\hat{y}$ direction both moving at velocities α_x and α_y, respectively. ∇^2 is the Laplacian operator with $\nabla^2 = \text{div}(\nabla u) = u_{xx} + u_{yy}$. We note that this is very similar to the previous chapter but now there are two dimensions of diffusion. β is still a positive real constant. This equation is still both second order and linear so we expect to be able to continue using finite difference methods.

5.1 Adapting 1D Methods to 2D

We can apply many of the same methods that we developed in 1D to a 2D spatial domain. Now, we discretize in two spatial directions, x and y, creating equally spaced partitions of width Δx and Δy. For simplicity, we set $\Delta x = \Delta y$, creating a regularly spaced grid. See figure 5.1.

We can also extend the same finite difference schemes to 2D. We restrict the values in $\vec{\alpha}$ to be positive, thus ensuring the advective direction is time-independent and always in the $\hat{x}$ and $\hat{y}$ directions. Thus, it is possible to apply the upwind scheme in much the same way as before, only this time in two dimensions. The subscript on α denotes the basis component of the advection direction. We also introduce an extended notation for indexing in two dimensions. j is the index

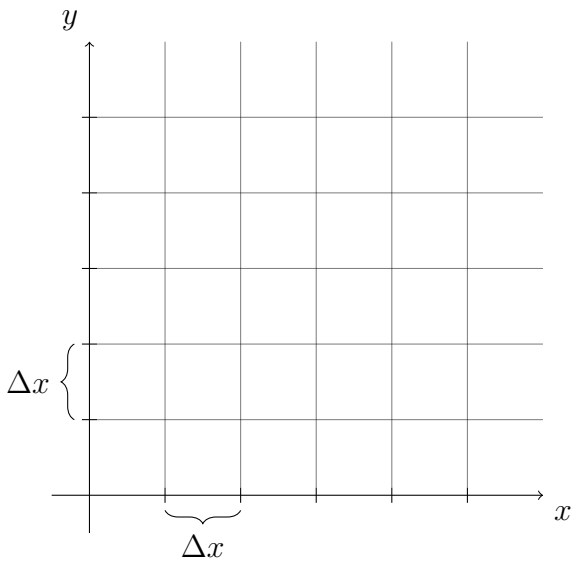

Figure 5.1: Discretizing the 2D spatial domain.

in the x-direction and i the y-direction.

$$\vec{\alpha} \cdot \nabla u = \frac{\alpha_x}{\Delta x}\left(U^k_{j,i} - U^k_{j-1,i}\right) + \frac{\alpha_y}{\Delta x}\left(U^k_{j,i} - U^k_{j,i-1}\right) \tag{5.2}$$

Much the same can be done with diffusion. Notice how the diffusion operator reduces to the addition of two 1D cases, one for each coordinate direction. This gives us a finite difference scheme widely known as a 5-point stencil for the 2D Laplacian:

$$\beta\nabla^2 u = \frac{\beta}{\Delta x^2}\left(U^k_{j-1,i} - 2U^k_{j,i} + U^k_{j+1,i}\right) + \frac{\beta}{\Delta x^2}\left(U^k_{j,i-1} - 2U^k_{j,i} + U^k_{j,i+1}\right). \tag{5.3}$$

In the same way as in section 4.2.1, we apply Forward Euler for the temporal derivative as before and we obtain an expression for the full 2D finite difference formulation

$$\begin{aligned}\frac{U^{k+1}_{j,i} - U^k_{j,i}}{\Delta t} = -\frac{\alpha_x}{\Delta x}\left(U^k_{j,i} - U^k_{j-1,i}\right) - \frac{\alpha_y}{\Delta x}\left(U^k_{j,i} - U^k_{j,i-1}\right) + \\ + \frac{\beta}{\Delta x^2}\left(U^k_{j-1,i} + U^k_{j,i-1} + U^k_{j+1,i} + U^k_{j,i+1} - 4U^k_{j,i}\right)\end{aligned} \tag{5.4}$$

The last step is to propagate this equation forward in time similarly to the 1D case with matrix-

vector operations.

5.2 Solving with Linear Algebra Methods

Solving the advection-diffusion equation in 2D is more difficult than in 1D for many reasons:

1. There are now derivatives in multiple spatial directions.

2. $\mathbf{U}^k$ is now a 2D array that contains the solution information. This is harder to deal with because it must be unpacked by rows or columns, operated upon, then repacked into the 2D array.

3. Boundary conditions and derivatives now possibly rely on different directions, which need to be represented in the unpacked array.

The solution $\mathbf{U}^k$ evaluated on an $N \times N$ grid of equally spaced points in the x and y-directions over the unit square, $[0, 1] \times [0, 1]$, is defined as follows by labeling each point in the domain.

$$\mathbf{U}^k = \begin{bmatrix} U_{1,1} & U_{1,2} & \cdots & U_{1,N} \\ U_{2,1} & U_{2,2} & & U_{2,N} \\ \vdots & & \ddots & \vdots \\ U_{N,1} & U_{N,2} & \cdots & U_{N,N} \end{bmatrix}$$

We want to apply something similar to our 1D derivative matrix $\mathbb{M} = [I - (\mathbb{D} - \mathbb{A})]$ from section 4.2 to each row of $\mathbf{U}^k$ in the x-direction and each column of $\mathbf{U}^k$ in the y-direction. As in the previous chapter, $\mathbb{D}$ remains the diffusion matrix and $\mathbb{A}$ remains the advection matrix. In order to do this, we want to be able to unpack the size $(N \times N)$ $\mathbf{U}^k$ into a vector of size $(1 \times N^2)$ by concatenating the rows. Finally, we then want to apply some new $\mathbb{M}_{2D}$, now size $N^2 \times N^2$, in order to find $\mathbf{U}^{k+1}$.

$$\mathbf{U}^k_{Row} = \left[U_{1,1} \quad U_{1,2} \quad \cdots \quad U_{1,N} \,\middle|\, U_{2,1} \quad \cdots \quad U_{2,N} \,\middle|\, \cdots \,\middle|\, U_{N,1} \quad \cdots \quad U_{N,N} \right] \tag{5.5}$$

5.2.1 Examining Structure of Finite Difference Matrices

After unpacking $\mathbf{U}^k$, we need to apply the derivative matrices in the x and y directions on the unpacking. This is a nontrivial task. First, we examine exactly how the different elements of $\mathbf{U}^k$ combine when calculating $\mathbf{U}^{k+1}$. For a small domain with indexed boundary values L, B, R, and T for left, bottom, right, and top; as well as corners, C, we examine the finite difference scheme for the central value in the domain.

$$\left[\begin{array}{c|ccc|c} C & T_1 & T_2 & T_3 & C \\ \hline L_3 & U_{1,1} & U_{1,2} & U_{1,3} & R_3 \\ L_2 & U_{2,1} & U_{2,2} & U_{2,3} & R_2 \\ L_1 & U_{3,1} & U_{3,2} & U_{3,3} & R_1 \\ \hline C & B_1 & B_2 & B_3 & C \end{array}\right]$$

$$\begin{aligned} U_{2,2}^{k+1} = U_{2,2}^k + \frac{\beta\Delta t}{\Delta x^2}\left[-4U_{2,2}^k + U_{2,3}^k + U_{3,2}^k + U_{2,1}^k + U_{1,2}^k\right] - \\ - \frac{\alpha_x\Delta t}{\Delta x}\left(U_{2,2}^k - U_{2,1}^k\right) - \frac{\alpha_y\Delta t}{\Delta x}\left(U_{2,2}^k - U_{3,2}^k\right) \end{aligned} \tag{5.6}$$

We want to be able to construct the $\mathbb{A}$ and $\mathbb{D}$ matrices to apply diffusion and advection operations onto $\mathbf{U}^k$ as in equation (4.19), only this time both will be size $N^2 \times N^2$ to capture the 2D behavior. We also define $\mathbb{A}_{2D}$ and $\mathbb{D}_{2D}$ to differentiate between these 1D and 2D advection and diffusion matrices. We introduce block matrices for this task to provide a visual indication of where an unpacked $\mathbf{U}_{Row}^k$ corresponds to the entries. This makes it easier when investigating where certain rows of a packed $\mathbf{U}^k$ correspond to regions of these block matrices. We also generalize equation (5.6) to a larger domain to better see the patterns formed in the following matrices. As in the 1D case, we delay looking at boundary conditions until later in this chapter.

$$\mathbb{A}_{2D} = \mathbb{A}_x + \mathbb{A}_y$$

$$= \frac{\alpha_x \Delta t}{\Delta x} \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right] + \frac{\alpha_y \Delta t}{\Delta x} \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{5.7}$$

$$\mathbb{D}_{2D} = \frac{\beta \Delta t}{\Delta x^2} \left[\begin{array}{ccc|ccc|ccc} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{array}\right] \tag{5.8}$$

The diagonal structures in the results form an interesting pattern that comes from applying the derivative operators in each direction. We now introduce a natural algebraic tool to encode this using tensor products of matrices which we introduce and define in the next section.

5.2.2 Introduction to Kronecker Products

Given matrices $\mathbb{A} \in \mathbb{R}^{n_1,m_1}$ and $\mathbb{B} \in \mathbb{R}^{n_2,m_2}$, the Kronecker product matrix $\mathbb{C} = \mathbb{A} \otimes \mathbb{B} \in \mathbb{R}^{N \times M}$ is defined entrywise as $C(i,j) = A(i_1,j_1)B(i_2,j_2)$, with $i = n_2(i_1 - 1) + i_2$ and $m_2(j_1 - 1) + j_2$, with $N = n_1 n_2, M = m_1 m_2$.

We provide an example to show how the above algorithm for forming these tensor product matrices results in the patterns that we are looking for in $\mathbb{A}_{2D}$ and $\mathbb{D}_{2D}$. Let

$$\mathbb{A} = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}, \ \mathbb{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using the above definition of the Kronecker product, we compute $\mathbb{A} \otimes \mathbb{B}$:

$$\mathbb{A} \otimes \mathbb{B} = \left[\begin{array}{ccc|ccc|ccc} -2*1 & -2*0 & -2*0 & 1*1 & 1*0 & 1*0 & 0*1 & 0*0 & 0*0 \\ -2*0 & -2*1 & -2*0 & 1*0 & 1*1 & 1*0 & 0*0 & 0*1 & 0*0 \\ -2*0 & -2*0 & -2*1 & 1*0 & 1*0 & 1*1 & 0*0 & 0*0 & 0*1 \\ \hline 1*1 & 1*0 & 1*0 & -2*1 & -2*0 & -2*0 & 1*1 & 1*0 & 1*0 \\ 1*0 & 1*1 & 1*0 & -2*0 & -2*1 & -2*0 & 1*0 & 1*1 & 1*0 \\ 1*0 & 1*0 & 1*1 & -2*0 & -2*0 & -2*1 & 1*0 & 1*0 & 1*1 \\ \hline 0*1 & 0*0 & 0*0 & 1*1 & 1*0 & 1*0 & -2*1 & -2*0 & -2*0 \\ 0*0 & 0*1 & 0*0 & 1*0 & 1*1 & 1*0 & -2*0 & -2*1 & -2*0 \\ 0*0 & 0*0 & 0*1 & 1*0 & 1*0 & 1*1 & -2*0 & -2*0 & -2*1 \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc|ccc} -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 \end{array}\right]$$

We also note that adding $A \otimes B + B \otimes A$ results in

$$\mathbb{A} \otimes \mathbb{B} + \mathbb{B} \otimes \mathbb{A} = \left[\begin{array}{ccc|ccc|ccc} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{array}\right]$$

Which is the same as $\mathbb{D}_{2D}$ from equation (5.8) up to a constant factor. This operation is called the Kronecker sum, denoted as $\mathbb{A} \oplus \mathbb{B} = \mathbb{A} \otimes I + I \otimes \mathbb{B}$. Here, it corresponds to applying the second derivative operators on each direction, then adding them. There are many well-known properties of this sum relating the eigenvalues and solution of linear systems for $\mathbb{A} \oplus \mathbb{B}$ to those of $\mathbb{A}$ and $\mathbb{B}$.

The same is true with $\mathbb{A}_{2D}$ from equation (5.7) although a slight adjustment must be made because we assume $\alpha_x \neq \alpha_y$ so $\mathbb{A}_{2D} = \mathbb{A}_x \oplus \mathbb{A}_y = \mathbb{A}_x \otimes I + I \otimes \mathbb{A}_y$ in order to correctly implement

the direction of advection. These results show that we can write the time advancement of $\mathbf{U}_{Row}^{k}$ again with a linear algebra framework by using Kronecker sums.

$$\mathbf{U}_{Row}^{k+1} = \left[I + \Delta t \left[(\mathbb{D} \oplus \mathbb{D}) - (\mathbb{A}_x \oplus \mathbb{A}_y)\right]\right] \mathbf{U}_{Row}^{k} \tag{5.9}$$

We rewrite the equation above with $\mathbb{M}_{2D}$, the 2D time advancement matrix of size $(n^2 \times n^2)$:

$$\mathbf{U}_{Row}^{k+1} = \mathbb{M}_{2D} \mathbf{U}_{Row}^{k} \tag{5.10}$$

5.3 Implementing Boundary Conditions

As in the previous section, we have neglected to consider boundary conditions until now. Let us go back to the domain from above using Dirichlet boundary conditions. Before we begin, we need to consider how our domain changes. Similar to the 1D case, we now know the values of $\mathbf{U}_{Row}^{k}$ on the boundaries so our $\mathbb{M}_{2D}$ matrix can be reduced from size $n^2 \times n^2$ to $(n-2)^2 \times (n-2)^2$. We call this new matrix $\mathbb{M}_{2D,D}$, representing a 2D matrix with Dirichlet boundary conditions. We define our boundaries as we did in the last section:

$$\left[\begin{array}{c|ccc|c} C & T_1 & T_2 & T_3 & C \\ \hline L_3 & U_{1,1} & U_{1,2} & U_{1,3} & R_3 \\ L_2 & U_{2,1} & U_{2,2} & U_{2,3} & R_2 \\ L_1 & U_{3,1} & U_{3,2} & U_{3,3} & R_1 \\ \hline C & B_1 & B_2 & B_3 & C \end{array}\right]$$

and this time examine the formulas for a corner, $U_{1,1}$, and an edge, $U_{1,2}$. We begin with $U_{1,1}$.

$$U_{1,1}^{k+1} = U_{1,1}^{k} + \frac{\beta\Delta t}{\Delta x^2}\left[-4U_{1,1}^{k} + U_{1,2}^{k} + U_{2,1}^{k} + L_3 + T_1\right] - \\ - \frac{\alpha_x}{\Delta x}\left(U_{1,1}^{k} - L_3\right) - \frac{\alpha_y}{\Delta x}\left(U_{1,1}^{k} - U_{2,1}^{k}\right)$$

We can remove any parts not dependent on U:

$$U_{1,1}^{k+1} = U_{1,1}^{k} + \frac{\beta\Delta k}{\Delta x^2}\left[-4U_{1,1}^{k} + U_{1,2}^{k} + U_{2,1}^{k}\right] + \frac{\alpha_x}{\Delta x}\left(U_{1,1}^{k}\right) + \frac{\alpha_y}{\Delta x}\left(U_{1,1}^{k} - U_{2,1}^{k}\right) + \\ + \frac{\beta\Delta t}{\Delta x^2}\left(L_3 + T_1\right) + \frac{\alpha_x}{\Delta x}\left(L_3\right) \tag{5.11}$$

Which as before leaves us with a term that is expressed in $\mathbb{M}_{2D}$ as well as terms that will be expressed with a vector containing boundary conditions. Continuing with $U_{1,2}$:

$$U_{1,2}^{t+\Delta t} = U_{1,2}^{t} + \frac{\beta\Delta t}{\Delta x^2}\left[-4U_{1,2}^{t} + U_{1,1}^{t} + U_{2,2}^{t} + U_{1,3}^{t}\right] + \frac{\alpha_x}{\Delta x}\left(U_{1,2}^{t} - U_{1,1}^{t}\right) + \frac{\alpha_y}{\Delta x}\left(U_{1,2}^{t} - U_{2,2}^{t}\right) + \\ + \frac{\beta\Delta t}{\Delta x^2}\left(T_1\right) \tag{5.12}$$

Due to our adoption of Dirichlet boundary conditions, we shrink the size of $\mathbb{M}_{2D}$ from $n^2 \times n^2$ to $(n-2)^2 \times (n-2)^2$. The reason for this is the same as last section when we reduced $\mathbb{M}$ from $n \times n$ to $(n-2) \times (n-2)$. When we specify the boundary conditions, we no longer need to solve for the boundaries and thus we can cut them out of our matrices, saving computing power. For the 1D case this has a marginal effect, reducing the size of $\mathbb{M}$ by a factor of n. For $\mathbb{M}_{2D}$, however, this reduces the size of $\mathbb{M}_{2D}$ by a factor of n^3.

Again, we are left with a term that can be expressed with the matrix $\mathbb{M}_{2D}$ and an additional boundary term, $\mathbf{B}_{2D,D}$. The 'D' specifies Dirichlet. In fact, every term on the boundary of the domain has boundary values associated with it. The interior terms do not require information about the boundaries and thus do not need adjustment with a boundary term. Finally, we can

express the complete time advancement with boundary conditions as:

$$\mathbf{U}_{Row}^{k+1} = \mathbb{M}_{2D}\mathbf{U}_{Row}^{k} + \mathbf{B}_{2D,D,Row} \tag{5.13}$$

$\mathbf{B}_{2D,D,Row}$ is the unpacked boundary vector that in unpacked in the same manner as $\mathbf{U}_{Row}^{k}$. We use an example with $n = 10$ to show how $\mathbb{M}_{2D,D}$ and $\mathbf{B}_{2D}$ are constructed. Figures 5.3 and 5.2 showcase the structure of $\mathbb{M}_{2D}$ and $\mathbf{B}_{2D}$, before unpacking it, respectively. In figure 5.2, notice as

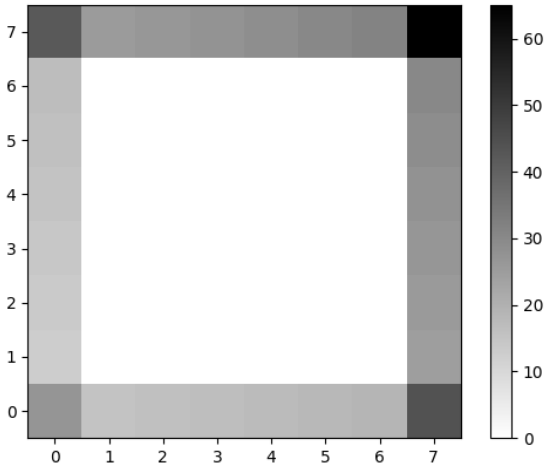

Figure 5.2: Structure of boundary conditions for a $\Delta x = 0.1$ or $n = 10$ domain.

the boundary conditions are Dirichlet, $n = 10 \rightarrow n = 8$ as the edges are known. The values on the boundaries change as the boundary values are now spatially dependent functions. Finally, the corners are larger in magnitude than their surroundings because, as in equation (5.11), there are more boundary terms to computer corner terms than edge terms. $\mathbf{B}_{2D,D}$ is unpacked by concatenating its rows so that it is length $(n-2)^2$ or in this case, length 64. After unpacking, it can be applied to $\mathbb{M}_{2D}$ in figure 5.3. Notice the dimensions are $8^2 \times 8^2 = 64 \times 64$ with the repeating structure. When the unpacked $\mathbf{B}_{2D,D,Row}$ is applied, each of the original rows interacts with the correspond-

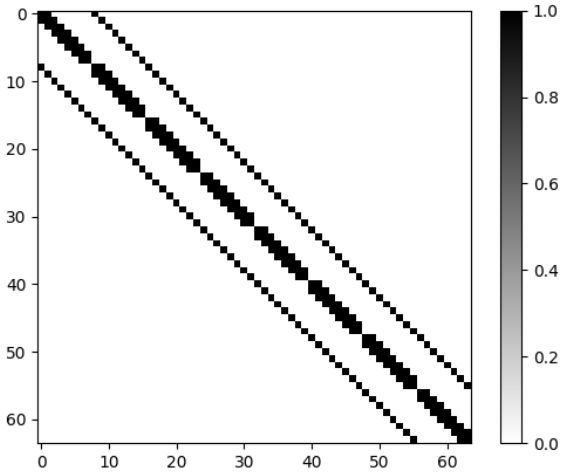

Figure 5.3: Structure of the matrix structure for the same problem as figure 5.2.

ing chunk on the diagonal. The terms in $\mathbb{M}_{2D}$ are Kronecker sums of tridiagonal matrices, which results in a $n \times n$ block tridiagonal matrix, where each diagonal block is itself tridiagonal, and each superdiagonal and infradiagonal term is also diagonal. This sparse structure is well known for the 5 point Laplacian stencil and this discretization of the advective term shares the same entries. The same methods as in section 4.3 can be used to apply Robin boundary conditions.

5.4 Checking Validity of Numerical Solutions

We can use the same method as section 4.4 where we check that our finite difference scheme approximates the steady state large t limit of the PDE, but this time we are working with the 2D elliptic problem:

$$0 = -(\alpha_x u_x + \alpha_y u_y) + \beta(u_{xx} + u_{yy}). \tag{5.14}$$

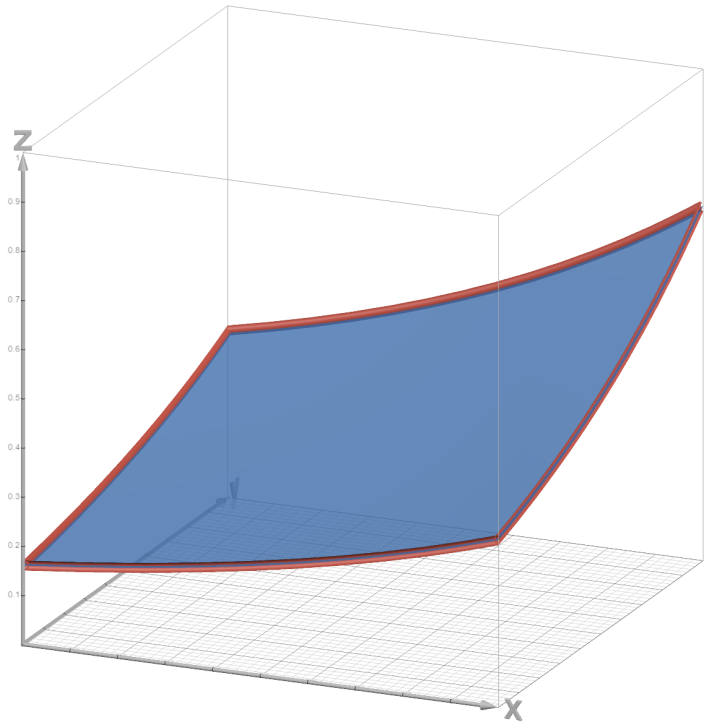

Figure 5.4: Analytical solution to the 2D elliptic problem.

We still assume this to be defined over the unit square to simplify calculations. We can use separation of variables to obtain the general solution for u.

$$u(x,y) = (C_1 + C_2 e^{\frac{\alpha_x}{\beta}x})(C_3 + C_4 e^{\frac{\alpha_y}{\beta}y}) \tag{5.15}$$

Where $C_1, C_2, C_3, C_4 \in \mathbb{R}$. Figure 5.4 is an example of such a solution. In this case, we use boundary conditions that are smooth across the boundary of the domain to avoid any discontinuities in the analytical solution. This is not strictly necessary, but makes the task much easier. Figure 5.4 is an example of smooth boundaries and the resulting well-behaved analytical solution. Each boundary in figure 5.4 is an exponential curve described by equation (5.15). The particular example in figure 5.4 is $\vec{\alpha} = \left\langle \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\rangle$ and $\beta = 0.5$.

We begin by examining our numerical solution to the elliptic problem to confirm that our

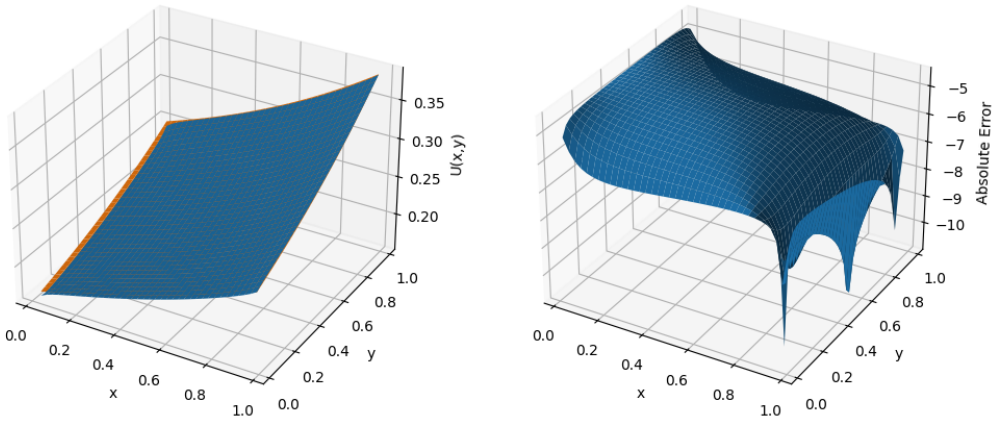

Figure 5.5: Numerical solution to the 2D elliptic Dirichlet problem with $n = 40$. The right plot has a log scale on the z-axis.

$\mathbb{M}_{2D,D}$ and $\mathbf{B}_{2D,D}$ are constructed correctly for the Dirichlet boundary conditions. Figure 5.5 shows the numerical solution to the 2D elliptic Dirichlet problem. On the left, the orange sheet is the analytical solution and the blue sheet is the numerical solution to the elliptic problem. The right plot showcases the logarithm of the absolute error between the two.

5.5 Conclusion and Extensions to 2D Diffusive Burgers' Equation

In this chapter we extended our 1D results to the 2D unit square. In doing so, we introduced new notation and most importantly new linear algebra operations that allowed us to compute derivatives in multiple directions over the domain. This chapter finishes the portion of this thesis where we consider purely linear PDEs and now we move on to our next building block, the diffusive Burgers' equation, our final step in producing a numerical framework for chemotaxis.

CHAPTER 6

2D DIFFUSIVE BURGERS' EQUATION

6.1 Introduction

Burgers' equation is a nonlinear PDE that exhibits advective behavior. It can be derived from the Navier-Stokes equations and is often used to model shocks in fluids. Solutions to Burgers' equation often exhibit discontinuities which come from its non-linearity. Discontinuities are difficult to model so we introduce a diffusive term to smooth the solution. The diffusive version is aptly named the diffusive Burgers' equation and it is a second order non-linear PDE. Depending on the magnitude of the diffusion term, solutions to diffusive Burgers' can look very similar to solutions of Burgers' equation with shock-like near discontinuities or if the diffusion parameter is very large, the solutions may behave more like the advection diffusion equations we have previously dealt with.

The diffusive Burgers' equation is similar to the advection diffusion equation we have covered in the previous chapters but it is also closely related to our transformed, non-dimensionalized model (2.10) from section 2.3. This makes it an appropriate final step in this thesis for establishing a framework for numerically modeling chemotaxis in two dimensional space. The Diffusive Burgers' equation is written

$$u_t = -\alpha \nabla \cdot \left(\frac{1}{2} u^2 \right) + \beta \Delta u. \tag{6.1}$$

In this chapter, we examine numerical solutions to the diffusive Burgers' equation in 1D before extending them–like we did with the advection diffusion equation–to 2D.

6.2 1D Diffusive Burgers' Equation

We start by looking at finite difference methods to solve diffusive Burgers' equation in 1D. From here, we will continue to generalize our methods to 2D. Notice in 1D:

$$-\alpha\nabla\cdot\left(\frac{1}{2}u^2\right) = -\alpha u u_x$$

There are two immediate options for how we could formulate this with finite differences depending on how we evaluate $-\alpha\nabla\cdot\left(\frac{1}{2}u^2\right)$. If we think of it as $-\alpha u u_x$, we find:

$$\frac{u_j^{t+1}-U_j^t}{\Delta t} = -\alpha U_j^t\frac{U_j^t-U_{j-1}^t}{\Delta x} + \beta\frac{U_{j-1}^t-2U_j^t+U_{j+1}^t}{\Delta x^2}. \tag{6.2}$$

If we express it as $F(u)_x$, where $F = 1/2u^2$, it takes the form:

$$\frac{U_j^{t+1}-U_j^t}{\Delta t} = -\frac{\alpha}{2}\frac{\left(U_j^t\right)^2-\left(U_{j-1}^t\right)^2}{\Delta x} + \beta\frac{U_{j-1}^t-2U_j^t+U_{j+1}^t}{\Delta x^2}. \tag{6.3}$$

We take a moment to introduce conservation laws. Conservation laws, such as those governing mass, momentum, and energy, are fundamental principles in physics. Numerical methods that accurately preserve these conservation laws ensure that important physical quantities remain consistent over time in the discrete solution, reflecting the behavior of the physical system.

When a numerical method fails to conserve these quantities, it may introduce artificial effects or errors into the solution, leading to non-physical behavior. For example, if a numerical method does not conserve mass when simulating fluid-flow, it may lead to nonphysical mass accumulation or depletion in certain regions.

Many PDEs are derived from conservation laws. We derived the chemotaxis model from a conservation law in chapter 2. For a method to be conservative, the total value of u over an interval can only change due to flux through the interval boundaries. Over the interval $[L, R]$ and for some function Φ representing the flux through the boundaries, a conservative equation must be able to be written in the form:

$$\int_L^R u(x, t + \Delta t)dx = \int_L^R u(x, t)dx + \int_t^{t+\Delta t} \Phi(L, t)dt - \int_t^{t+\Delta t} \Phi(R, t)dt$$

This extends to a discretization of an interval where $R = L + \Delta x$ and thus is also applicable to numerical solutions.

The second method is the conservative form of Burgers' equation [18]. Continuing with the second method:

$$U_j^{t+1} = U_j^t - \frac{\alpha \Delta t}{2\Delta x}\left[\left(U_j^t\right)^2 - \left(U_{j-1}^t\right)^2\right] + \frac{\beta \Delta t}{\Delta x^2}(U_{j-1}^t - 2U_j^t + U_{j+1}^t). \tag{6.4}$$

This is an acceptable way to deal with diffusive Burgers' equation when we keep the advection direction constant, but this method will fail if the direction of advection changes. In section 4.1.2, we determined that upwind methods for solving the advection equation only worked for direction of fluid flow. Our model for chemotaxis does not assume constant direction advection and using this finite difference scheme would not work. In order to fix this, we introduce another method for solving Burgers' equation for a general advection direction: the Lax-Wendroff scheme [19]. This

method is a well known and standard method of solution to the diffusive Burgers' equation.

$$\begin{aligned} U^{t+\Delta t} = U^t - \frac{\Delta t}{2\Delta x}\left[F(U^t_{j+1}) - F(U^t_{j-1})\right] + \\ + \frac{\Delta t^2}{2\Delta x^2}\left[A_{j+1/2}(F(U^t_{j+1}) - F(U^t_j)) - A_{j-1/2}(F(U^t_j) - F(U^t_{j-1}))\right] \end{aligned} \tag{6.5}$$

Here, $A_{j\pm1/2} = \frac{1}{2}\left(U^t_j + U^t_{j\pm1}\right)$. $F(u) = \frac{1}{2}u^2$ as above. We can add the diffusive aspect of the equation to complete our Finite Difference scheme for diffusive Burgers' as:

$$\begin{aligned} U^{t+\Delta t} = U^t - \frac{\Delta t}{2\Delta x}\left[F(U^t_{j+1}) - F(U^t_{j-1})\right] + \\ + \frac{\Delta t^2}{2\Delta x^2}\left[A_{j+1/2}(F(U^t_{j+1}) - F(U^t_j)) - A_{j-1/2}(F(U^t_j) - F(U^t_{j-1}))\right] + \\ + \frac{\beta\Delta t}{\Delta x^2}\left(U^t_{j-1} - 2U^t_j + U^t_{j+1}\right). \end{aligned} \tag{6.6}$$

We now have a 1D finite difference method for the diffusive Burgers' equation that uses two vectors, $\mathbf{U}^k$ and $F(\mathbf{U}^k)$ to evolve the solution. We clarify our definition of $F(\mathbf{U}^k)$ to be $F : \mathbb{R}^m \to \mathbb{R}^m$ such that $F(\mathbf{U}^k) = \left\langle F(U^k_1), F(U^k_2), ..., F(U^k_n)\right\rangle$. This allows us to element-wise operate on our solution vector $\mathbf{U}^k$.

6.3 Solving with Linear Algebra Methods

Again, we would like to represent this as a matrix equation. However, due to the nonlinear $F(\mathbf{U}^k)$ and time dependent $A(t, \mathbf{U}^k)$, we will have to compute time dependent matrices at each time step. Below, we make use of the fact that when a diagonal matrix multiplies a dense matrix, the rows of the dense matrix are multiplied by the corresponding diagonal entry in the diagonal matrix when computing the Δt^2 term from equation (6.6). When the values of a matrix are dependent on $\mathbf{U}^k$ at a particular timestep, k, the matrix is represented with a superscript k. For example $\mathbb{\Gamma}^k$ is a matrix,

$\mathbb{\Gamma}$ that depends on $\mathbf{U}^k$ at a time k. At each timestep, we must recalculate $\mathbb{\Gamma}^k$, or any other time dependent matrix.

$$\mathbf{U}^{k+1} = \Delta t \left[-\mathbb{\Gamma}^k + \Delta t (\mathbb{\Omega}_+^k \mathbb{\Lambda}_+^k - \mathbb{\Omega}_-^k \mathbb{\Lambda}_-^k) \right] + \left[I + \Delta t \mathbb{D} \right] \mathbf{U}^k + \mathbf{B}^k \tag{6.7}$$

Here, I and $\mathbb{D}$ are the identity and diffusion matrices used for the advection-diffusion equation. b^t is a boundary vector as before but now has time dependencies due to the time dependencies of the matrix element. We include the definitions of in $\mathbb{\Gamma}^k$, $\mathbb{\Omega}_-^k$, $\mathbb{\Omega}_+^k$, $\mathbb{\Lambda}_+^k$, and $\mathbb{\Lambda}_-^k$ in the appendix. $\mathbf{B}^k$ is computed the same way as in section 4.3, but in this case the terms are more complicated. We go through each term in equation (6.7) and evaluate where the boundaries are not represented by the matrices. Then, we sum up the contribution in each term. The process differs between different boundary conditions.

6.4 Extending to 2D Domain

2D diffusive Burgers' equation is similar to the 1D case.

$$\begin{aligned} u_t &= -\alpha \nabla \cdot F(u) + \beta \Delta u \\ &= -\alpha \left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y} \right) F(u) + \beta \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u \end{aligned} \tag{6.8}$$

Similarly to the linear advection-diffusion case, we make use of Kronecker products to compute a $n^2 \times n^2$ formulation for U over the domain. We expand each of the matrices derived in the previous section in terms of the familiar Kronecker product. We then express the result in a simpler form

using Kronecker sums, as defined in section 5.2.2

$$\begin{aligned}
\mathbb{\Gamma}^k &\rightarrow \mathbb{\Gamma}^k \oplus \mathbb{\Gamma}^k, \\
\mathbb{\Omega}_+^k \mathbb{\Lambda}_+^k &\rightarrow \mathbb{\Omega}_+^k \mathbb{\Lambda}_+^k \oplus \mathbb{\Omega}_+^k \mathbb{\Lambda}_+^k, \\
\mathbb{\Omega}_-^k \mathbb{\Lambda}_-^k &\rightarrow \mathbb{\Omega}_-^k \mathbb{\Lambda}_-^k \oplus \mathbb{\Omega}_-^k \mathbb{\Lambda}_-^k, \\
\mathbb{D} &\rightarrow \mathbb{D} \oplus \mathbb{D}.
\end{aligned} \tag{6.9}$$

We reintroduce the 'Row' subscript that we used in section 5.2 to again represent the square domain unpacked by rows as detailed in equation (5.5). We also include $\mathbf{B}_{Row}^k$ representing the 2D boundary conditions unpacked in the same manner. We use the same methods as in the 2D advection diffusion equation to extend the 1D results in equation (6.7): expand the equation and substitute the results of (6.9) into (6.7),

$$\begin{aligned}
\mathbf{U}_{Row}^{k+1} =& \Delta t \left[\mathbb{\Gamma}^k \oplus \mathbb{\Gamma}^k\right] + \Delta t^2 \left[\mathbb{\Omega}_+^k \mathbb{\Lambda}_+^k \oplus \mathbb{\Omega}_+^k \mathbb{\Lambda}_+^k\right] - \Delta t^2 \left[\mathbb{\Omega}_-^k \mathbb{\Lambda}_-^k \oplus \mathbb{\Omega}_-^k \mathbb{\Lambda}_-^k\right] + \\
&+ \left[I + \Delta t \left(\mathbb{D} \oplus \mathbb{D}\right)\right] \mathbf{U}_{Row}^k + \mathbf{B}_{Row}^k.
\end{aligned} \tag{6.10}$$

Evolving equation (6.10) is considerably more computationally expensive than evolving the 2D advection diffusion equation. This is due to the time dependent matrices that must be computed at each time step. In addition, the boundary vector will change as well due to the inherent coupling with the matrices.

CHAPTER 7

SUMMARY AND FUTURE WORK

We began by exploring the implications of accuracy, stability, and convergence while developing finite difference methods for the 1D advection-diffusion equation. Extending our approach to a 2D Euclidean space, we devised methods for evolving the 2D advection-diffusion equation using linear algebraic techniques. Subsequently, we established a framework for numerically solving the diffusive Burgers' equation across a 2D square domain. Our model lays the foundations to numerical simulations to conservation laws with the hope to implement similar conservation laws for the tumor angiogenesis model.

The continued refinement of our model hinges on further analyzing the behavior of the term $\nabla\cdot(uv)$. A deeper analytical grasp of this term will enable us to refine the methods developed in this thesis for application in the tumor angiogenesis model. Additionally, we intend to explore the 1D system under a broader range of initial conditions and diffusion-to-growth rate ratios, potentially uncovering new relationships within the 1D model, particularly with respect to the parameter α on the solutions of the model. Finally, we desire to extend our results to curved, 2D surfaces to better approximate organ surfaces.

REFERENCES

[1] J. Adler, "Chemotaxis in bacteria," *Science*, vol. 153, no. 3737, pp. 708–716, 1966.

[2] K. E. Yong, "A mathematical model of the interactions between pollinators and their effects on pollination of almonds," *University of Iowa*, 2012.

[3] J. D. Murray, *Mathematical Biology I. An Introduction* (Interdisciplinary Applied Mathematics), 3rd ed. New York: Springer, 2002, vol. 17.

[4] M. A. J. Chaplain and A. M. Stuart, "A model mechanism for the chemotactic response of endothelial cells to tumour angiogenesis factor," *Math. Med. Biol.*, vol. 10, no. 3, pp. 149–168, 1993.

[5] A. Ruddell, A. Croft, K. Kelly-Spratt, M. Furuya, and C. Kemp, "Tumors induce coordinate growth of artery, vein, and lymphatic vessel triads," vol. 354, May 2014.

[6] P. Carmeliet, "Angiogenesis in health and disease," *Nat Med.*, 2003.

[7] M. D. Lazova, T. Ahmed, D. Bellomo, and T. S. Shimizu, "Response rescaling in bacterial chemotaxis," *Proceedings of the National Academy of Sciences*, 2011.

[8] D. A. H. II *et al.*, "Biologically-based mathematical modeling of tumor vasculature and angiogenesis via time-resolved imaging data," *Cancers*, 2021.

[9] H. Zhao and L. Zhu, "Dynamic analysis of a reaction-diffusion rumor propagation model," *International Journal of Bifurcation and Chaos, Volume 26, Issue 6, id. 1650101-52*, 2016.

[10] A. Hasan, N. Rodriguez, and L. Wong, "Transport and concentration of wealth: Modeling an amenities-based-theory," *Chaos*, vol. 30, no. 5, 2020.

[11] C. S. Patlak, "Random walk with persistence and external bias," *Bulletin of Mathematical Biophysics*, 1953.

[12] E. F. Keller and L. A. Segel, "Model for chemotaxis," *Journal of theoretical biology*, vol. 30, no. 2, pp. 225–234, 1971.

[13] P. Fuster Aguilera, "Qualitative analysis of a pde model for chemotaxis with logarithmic sensitivity and logistic growth," *Department of mathematics, Tulane University.*, 2021.

[14] B. Sleeman and H. Levine, "Partial differential equations of chemotaxis and angiogenesis," *Mathematical methods in the applied sciences*, vol. 24, no. 6, pp. 405–426, 2001.

[15] T. Li, R. Pan, and K. Zhao, "Global dynamics of a hyper- bolic parabolic model arising from chemotaxis," *SIAM Journal on Applied Mathematics*, 2012.

[16] W. Strauss, *Partial Differential Equations: An Introduction*. Wiley, 2007.

[17] R. Burden and J. Faires, *Numerical Analysis*. Cengage, 2011.

[18] R. J. LeVeque, *Numerical methods for conservation laws*. Springer, 1992, vol. 214.

[19] P. Lax and B. Wendroff, "Systems of conservation laws," *Los Alamos National Laboratory*, 1959.

Appendices

Matrices for 1D Diffusive Burgers' Equation

$$\mathbb{\Gamma}^k = \frac{1}{2\Delta x}\begin{bmatrix} 0 & F(U_2^k) & 0 & 0 & \cdots & 0 \\ -F(U_1^k) & 0 & F(U_3^k) & 0 & \cdots & 0 \\ 0 & -F(U_2^k) & 0 & F(U_4^k) & & 0 \\ 0 & 0 & -F(U_3^k) & 0 & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & F(U_n^k) \\ 0 & 0 & 0 & 0 & -F(U_{n-1}^k) & 0 \end{bmatrix}$$

$$\mathbb{\Lambda}_+^k = \frac{1}{2\Delta x^2}\begin{bmatrix} -F(U_1^k) & F(U_2^k) & 0 & 0 & \cdots & 0 \\ 0 & -F(U_2^k) & F(U_3^k) & 0 & \cdots & 0 \\ 0 & 0 & -F(U_3^k) & F(U_4^k) & & 0 \\ 0 & 0 & 0 & -F(U_4^k) & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & F(U_n^k) \\ 0 & 0 & 0 & 0 & 0 & -F(U_n^k) \end{bmatrix}$$

$$\mathbb{\Lambda}_-^k = \frac{1}{2\Delta x^2}\begin{bmatrix} F(U_1^k) & 0 & 0 & 0 & \cdots & 0 \\ -F(U_1^k) & F(U_2^k) & 0 & 0 & \cdots & 0 \\ 0 & -F(U_2^k) & F(U_3^k) & 0 & & 0 \\ 0 & 0 & -F(U_3^k) & F(U_4^k) & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & -F(U_{n-1}^k) & F(U_n^k) \end{bmatrix}$$

$$\mathbb{\Omega}_{+}^{k} = \frac{1}{2}\text{diag}\left\{\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & & 0 \\ 0 & 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{U}^{k} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right\}$$

$$\mathbb{\Omega}_{-}^{k} = \frac{1}{2}\text{diag}\left\{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & & 0 \\ 0 & 0 & 1 & 1 & \ddots & 0 \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \mathbf{U}^{k} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right\}$$

Acknowledgements

I'd like to sincerely thank Dr. Padi Fuster and Dr. Eduardo Corona for their invaluable support over the past years as I have been working on this project. Their guidance and mentorship enabled me to tackle this problem. I'd also like to thank Dr. James Rickards for his mentorship over the past three years and give a shout out for him being responsible for an astounding 30% of my collegiate mathematics education. In a similar vein, I'd like to thank Zach Jordan for his friendship over the past four years and many classes. Finally, I'd like to thank my friends and family for their support and encouragement through the many tough moments, I couldn't have done it without y'all.