SIMULATING SPATIAL PARTIAL DIFFERENTIAL EQUATIONS WITH CELLULAR AUTOMATA

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Brian Paul Strader

December 2008

SIMULATING SPATIAL PARTIAL DIFFERENTIAL EQUATIONS WITH

CELLULAR AUTOMATA

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Brian Paul Strader

December 2008

Approved by:

Keith Evan Schubert, Chair, Department of Computer Science and Engineering	Date
George Georgiou	
Ernesto Gomez	

ABSTRACT

Spatial partial differential equations are commonly used to describe systems of biological entities, such as patterns of desert vegetation. These equations can be transformed into cellular automata models, which have the benefit of being easily simulated, highly parallelizable, and change the perspective of the model from a global view to a local view. In this thesis I propose two methods for transforming a subset of partial differential equations into cellular automata models. The transformations are accomplished using discretization methods and the Forward and Backward Euler's methods.

Stability and convergence for the new cellular automata models are then explored for a subset of the models only containing linear terms. First the theoretical bounds of stability of the models are found using the Z-transform. Multiple simulations are then used to map out the areas where the cellular automata models will converge to stable values based upon how time and space are discretized. Stiffness of the cellular automata models is also explored to determine whether or not it has an impact upon stability. From this information, I provide a set of guidelines about what parameters to pick, with respect to discretization. These guidelines will help a biologist using one of the models to ensure that the simulations will converge to stable values and that the simulations will run quickly.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Keith Schubert for his time, knowledge, and patience while guiding me during this past year. I would also like to thank my committee members, Dr. George Georgiou and Dr. Ernesto Gomez, as well as Jane Curnutt, whose research my thesis is based upon, and my graduate coordinator Dr. Josephine Mendoza. I would also like to thank my parents Raleigh and Rene Strader as well as my brother Matthew Strader who had the unfortunate task of being my editor.

DEDICATION

This work is dedicated to my grandmother Carol Strader and my grandfather Alphonse Pirot, who have inspired me with their quiet strength and determination.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Purpose

Throughout the history of computer science, different models and machines have been used to determine what problems can be solved and how they can be measured. Finite automata, push-down automata, and Turing Machines describe by their very nature the limits of computability. Their usefulness lies in the fact that the most complex CPUs, systems with terabytes of RAM, can be theoretically bound by a piece of infinite tape and a set of states. In this thesis I investigate another model relationship, cellular automata and spatial partial differential equations.

Spatial partial differential equations occur all over in the natural world. They have been used in biology to model the behavior and patterns of organisms. The problem with partial differential equations is that while they may naturally fit the situation, they can be mathematically complex and difficult to solve. Cellular automata on the other hand use very simple mathematical rules, usually addition, subtraction, and conditional statements in order to create complex results. There are several indications, as shown throughout this introduction, that cellular automata are related to differential equations. In my thesis I will clearly define this relationship and show how an important subset of spatial differential equations can be transformed into cellular automata. I will also analyze what discretization sizes of space and time will allow the cellular automata to converge to a stable solution in the quickest possible simulation time.

1.2 Significance

If such rules could be derived, to allow for the creation of a cellular automata from a differential equation, then it leads one to believe that the opposite may also be possible. This means that a set of rules might be created which can turn a cellular automata into a generalized differential equation. If this is the case, then this will give biologists a powerful tool in describing the natural world. By creating a simple set of rules, complex mathematical equations could be derived, saving scientists time and effort to research their field rather than being bogged down into algebraic equations.

Such a tool would be able to take an image of a biological pattern and try to connect it with a particular cellular automata. The cellular automata could then be translated into a differential equation, which a biologist could then use to model the biological pattern. My thesis would be a first step in creating the groundwork for such a tool.

1.3 Findings

In this thesis I will show the following:

- Using a literature review of biological partial differential equations as a guide, a general formula was derived to encompass most of the important aspects of these equations. This is shown in Sections 3.2 and 3.3 on pages 18 and 21.

- Spatial partial differential equations can be converted into and simulated by cellular automata using approximations and Backward and Forward Euler's Methods, specifically in the case of the general biological formula found in the previous item. This is shown in Section 3.4 on page 21.

- Using the Z-transform method on the newly constructed cellular automata model, theoretical boundaries of stability for the model were found. This is shown for the Forward Euler's model in Section 4.1.1 on page 26 and for the Backward Euler's model in Section 4.1.2 on page 29.

- Convergence maps with respect to the parameters h_x and h_t (amounts in space and time the partial differential equations are discretized by) were created showing where the new cellular automata model converges and diverges. The theoretical boundaries of stability derived from the Z-transform are shown to closely match the shape of the area of convergence. This also proves that the size of the discretization for time and space can affect whether the new cellular automata model will converge or diverge. The convergence maps are shown in Section 5.1 on page 34 and the measurement of error between the theoretical and actual convergent boundaries are shown in Sections 5.2 and 5.3 on pages 38 and 46.

- Within the important area of convergence, the time to convergence is shorter the closer the h_x and h_t parameters are to the lower boundary of the convergence area. This is shown in Section 5.4 on page 49.

- A set of guidelines were created that can be used to determine the optimum parameters h_x and h_t for convergence and simulation speed. This is shown in Section 5.5 on page 50.

- An argument can be made that stiffness in not a problem for a portion of the convergence area. This is shown in Appendix B on page 88.

2. LITERATURE REVIEW

In this chapter I will conduct a general survey of research that will have bearing upon my topic. Section 2.1 on page 5 will briefly mention and cite the papers that provide background material for this thesis. Section 2.2 on page 6 provides examples of differential equations that describe biological processes including Population Growth (Section 2.2.1), Vegetation Patterns (Section 2.2.2), and Morphogenesis (Section 2.2.3). Section 2.3 on page 10 discusses how partial differential equations can be solved using discretization and approximation. Section 2.4 on page 11 explains how cellular automata models work and their benefits (Section 2.4.1), their applications (Section 2.4.2), their connections to differential equations and other models (Section 2.4.3), and lastly other attempts to transform differential equations to cellular automata (Section 2.4.4).

2.1 Background

There are several different examples of biological processes that are modeled by spatial differential equations. Several articles have been written concerning the growth patterns of vegetation in the desert[13] [6]. The biomass density of the vegetation is modeled by a differential equation that takes into account dryness of the soil and mortality of the plants. Turing also worked with modeling biological and chemical process with partial differential equations, explicitly in how morphogens move throughout cells during morphogenesis [12].

The study of cellular automata really took off with Conway's "game of life" [3]. Currently, a source that contains a wide range of research on cellular automata is Stephen Wolfram's *A New Kind of Science* [15]. It includes such topics as computability, modeling nature with cellular automata, and trying to generally classify types of cellular automata.

The basis for my thesis is work done by my advisor and several colleagues at CSUSB on *Patterned Growth in Extreme Environments* [2]. The paper details how several patterns created within nature that can be modeled by simple cellular automata rules. The article, however, did not define any clear rules on creating the cellular automata model from the patterns, which is what I will investigate in my research. Other articles such as [4] did make some conversions rules from differential equations to cellular automata, although somewhat specific to their particular situation.

2.2 Biology and Differential Equations

2.2.1 Population Growth

One basic area where it is easy to see how biology is related to differential equations is population growth. If P is the size of a population, then the growth rate of the population is described by the equation:

$$\frac{dP}{dt} = kP \tag{2.1}$$

where k is a growth constant. The equation itself is common sense, the rate a population of organisms will grow depends upon the current population size. What can make differential equations tricky is solving for a variable like P without having any derivatives left within the equation. In this case it is trivial, by simply separating the variables, placing all the elements of t on one side and all the elements containing P on the other and then integrating [11].

$$\int \frac{dP}{P} = \int k dt \tag{2.2}$$

$$P = Ae^{kt} \tag{2.3}$$

Here A is some arbitrary constant of integration. The following examples however cannot be solved so simply.

2.2.2 Vegetation Patterns

Certain patterns of vegetation within the deserts of Niger and Israel can be modeled by partial differential equations [13]. Where n(x,t) computes the biomass density and w(x,t) computes the water density:

$$\frac{\partial n}{\partial t} = \frac{yw}{1+\sigma w}n - n^2 - \mu n + \nabla^2 n \tag{2.4}$$

$$\frac{\partial w}{\partial t} = p - (1-\rho n)w - w^2 n + \delta\nabla^2(w - \beta n) - v\frac{\partial(w-\alpha n)}{\partial x} \tag{2.5}$$

In equation 2.4, $\frac{yw}{1+\sigma w}n$ describes plant growth with w standing for dry soil, $-\mu n$ describes mortality and being eaten by herbivores, and $-n^2$ accounts for "saturation due to limited nutrients." For equation 2.5, p represents precipitation, $(1-\rho n)w$ represents evaporation, and $-w^2 n$ represents transpiration.

According to these equations, as p (precipitation) varies, so do the patterns of the plants. Low precipitation leads to patches or spots of vegetation, which occurs because the plants draw water form the areas around them to absorb enough water. Higher precipitation levels create an interesting striped pattern that appears maze like, because they draw less water from surrounding areas. Even higher levels create uniform coverage with holes, similar in size to the patches of vegetation. These states of vegetation, i.e. spotted, striped, uniform with holes, bare, and completely uniform, are relatively stable. The fact that the states are stable is used by the authors to explain desertification. If vegetation in the spotted state does not receive enough precipitation over a period of time, the vegetation may fall to the completely bare state. Once in this lower stable state, one large rainfall will not be able to bring the vegetation back to the spotted vegetation state [13].

Several simulations were run using the vegetation models and several interesting facts were confirmed. There are certain values of p (precipitation) where two stable states can coexist. For example, simulations were run with the lower half receiving enough p for the spotted state and the upper half receiving p for the striped state. The result was a system that had the two patterns mix in the middle, especially if perturbations from the initial uniform vegetative state were allowed [6].

2.2.3 Morphogenesis

Alan Turing explored modeling biological patterns in morphogenesis, which are caused by morphogens. Morphogenesis is a category in developmental biology concerning how cells form into structures. Morphogens are those molecules or substances that direct the change in cells during morphogenesis. Morphogenesis attempts to describe processes such as how a clump of zygote cells can form into specific structures, creating an embryo. Chemistry is also an important factor in morphogenesis, because it is important to understand how the rate and ability for morphogens to diffuse through cells depends upon chemical reactions. Morphogenesis can also be compared to the previous vegetation sample, in that for both models a change in conditions (precipitation or diffusion of morphogens) can take a homogeneous state (bare soil or zygote cells) into complex structural patterns (labyrinth vegetation patterns or embryonic biological structures) [12].

Turing created a model using differential equations for a ring of N homogeneous cells, using X and Y to represent the amounts of 2 morphogens. A subscript r indicates the amount of X or Y in a particular cell r. This results in the two following formulas:

$$\frac{dX_r}{dt} = f(X_r, Y_r) + \mu(X_{r+1} - 2X_r + X_{r-1}) \tag{2.6}$$

$$\frac{dY_r}{dt} = g(X_r, Y_r) + \nu(Y_{r+1} - 2Y_r + Y_{r-1}) \tag{2.7}$$

For X and Y, the rate of change that is dependent upon chemical reactions is depicted receptively through $f(X_r, Y_r)$ and $g(X_r, Y_r)$. The rate of change also depends upon the diffusion of X and Y to the cells to the left or right in the ring of cells. This is depicted by the functions by the functions $\mu(X_{r+1} - 2X_r + X_{r-1})$ and $\nu(Y_{r+1} - 2Y_r + Y_{r-1})$.

Turing says that his simplified model, going from a homogeneous state to patterns, is not very helpful, because in the real world one generally starts with a pattern and has it evolve to another pattern. Although one would not be able to model every pattern to another, digital computers might be able to aid in identifying the change from certain special sub cases of patterns to other patterns.

2.3 Solving Partial Differential Equations

My research will be primarily dealing with partial differential equations because they are the most difficult to solve and techniques used to solve partial differential can be used on ordinary differential equations as well. The methods to solve partial differential equations that I will examine will be approximation methods, which discretize the equation. This will help with converting differential equations to cellular automata and back since cellular automata have rules applied to discrete areas.

One technique is to discretize the partial differential so that it is reduced to an ordinary differential equation. An example showing how this can be done is in [9] using the following wave equation:

$$\frac{1}{c^2} u_{tt} = u_{xx} \tag{2.8}$$

Say I want to solve the equation at points for x: $x_0, x_1, ...x_i, ...$ where h will be the difference between x_i and x_{i+1}. I want to replace u_{xx} to remove the variable x from the equation and substitute it with x_i. To do this I will use the three point formula, which approximates derivatives:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{2.9}$$

The second derivate can be approximated by replacing f(x) with f'(x) since the second derivative is the derivative of the first derivative, and also using half of h.

I then use the three point formula and replace f(x) with f'(x). This gives the following formulas once the substitution is made:

$$f''(x) = \frac{f'(x+\frac{h}{2}) - f'(x-\frac{h}{2})}{h} \tag{2.10}$$

$$= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{2.11}$$

I can now use this formula to replace u_{xx} in the wave equation. Also note that $x + h = x_{i+1}$ and $x - h = x_{i-1}$:

$$\frac{1}{c^2}u_{tt}(t, x_i) = \frac{u(t, x_{i+1}) - 2u(t, x_i) + u(t, x_{i-1})}{h^2} \tag{2.12}$$

$$u''(t, x_i) = \frac{-2c^2}{h^2}u(t, x_i) + \frac{c^2}{h^2}(u(t, x_{i+1}) + u(t, x_{i-1})) \tag{2.13}$$

So now I have a set of ordinary differential equations for each x_i, which can be approximated by several methods. Equation 2.12 implies a possible link between partial differential equations and cellular automata. The result for x_i is determined partially by x_{i-1} and x_{i+1}, the points to the left and the right of the point I am calculating. As you will see, the values in cellular automata are usually based upon the values of their neighbors.

2.4 Cellular Automata

2.4.1 Definition

Cellular automata (CA) are simple models that create surprisingly complex results. A CA is composed of a grid of cells. Each cell could contain a variety of things but usually it contains a number or the cell is simply filled or empty. The value of the cell is based upon a set of rules, which are usually based upon the neighboring cells. A simulation using a CA begins with an initial state with some cells having values while others are empty, having the equivalent value of zero. At each time period the set of rules is applied to each cell to see what the new value of each cell will be. Over many time periods the values within the cells will usually form some pattern, although it may not be uniform.

A simple example of a CA that displays complex results is the original automata that helped to spark current interest in subject, the "game of life". In the "game of life" each cell is either blank or filled in. A filled cell represents a living organism, hence the name the "game of life." Each cell has eight neighbors, those square cells that immediately surround the cell, horizontally, vertically, or diagonally. The rules to generate the next time period are as follows:

1. A cell that is filled will survive to the next round if it has two or three neighbors that are filled.

2. A cell that is filled with four or above filled neighbors will die from over population, or it will die from isolation if it has one or less filled neighbors.

3. A cell that is empty can be filled in if exactly three neighbors are filled, giving birth to a new organism.

This simulation game is played by creating an initial state of filled cells that create certain outcomes, such as stable cells that will stay filled every turn, or blinkers, cells that continually loop in a cycle of filled and empty [3].

The cellular automata model has several benefits. CA are not only simple mathematically, but they also are easy to simulate because they are already discrete. These simulations are also easily parallelizable. Each node in a distributed computing network can simulate a block of cells and pass results to each other about cell neighbors. The "game of life" in fact is a common problem used to introduce parallel and distributed computing to students. The ability to parallelize a problem or simulation is becoming a paramount concern as grid computing is now used to compute large simulations. Another benefit of CA is that they provide a local view to a problem. Cellular automata are constructed in terms of how a single cell interacts with its neighbor cells over time as opposed to how the overall pattern can be manipulated.

2.4.2 Applications

Ironically simple variations of Conway's "game of life" have been shown to create similar patterns to actual organisms. The game as stated somewhat describes the growth of desert vegetation in section 2.2.2 on page 7. The plants need some biomass around them to help them grow, meaning they need some neighbors to continue to live. Over population will occur if there are too many surrounding plants for the level of precipitation, killing some plants off. Changing the number of neighbors it takes to kill a cell in the "game of life" is like changing the level of precipitation the system gets. More precipitation allows for more neighbors to coexist and less neighbors coexist when water is running low in the area.

Another version of the "game of life" used to model biological patterns are that of Cyanobacteria, which helped to create oxygen in the atmosphere in Earth's ancient past [2]. The rules used are the same as the "game of life" except that if the cell has seven or more neighbors, it is killed rather than four or more. Also there is no death by isolation. Additionally an extra rule of random death is added, where there is a

10% chance that a cell will randomly die and be empty in the next time period. These rules will create an astonishingly similar pattern to the patterns left by Cyanobacteria as fossils, which are shown in Figure 2.1 The top cellular automata pattern in the figure was created in five time steps and the bottom in forty time steps.[2].

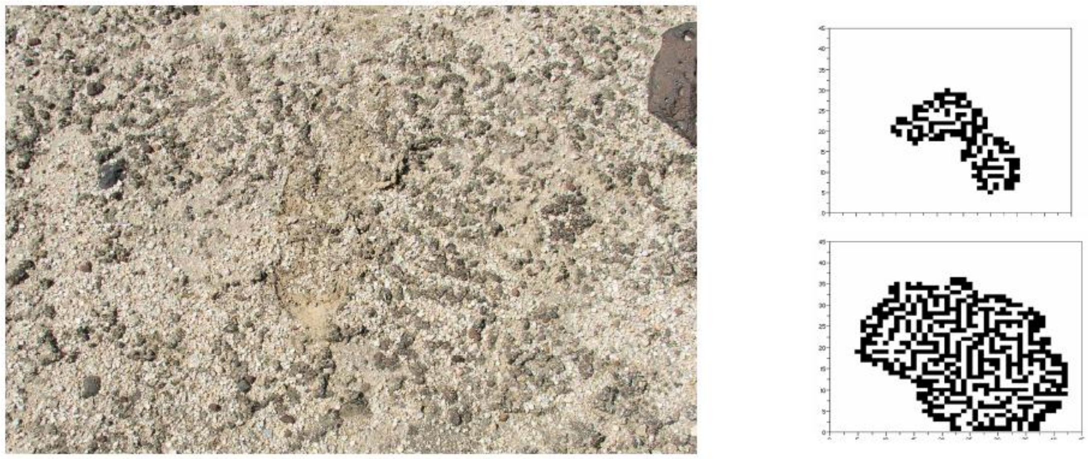

Fig. 2.1: Images of Cyanobacteria fossils and two patterns created by cellular automata.

Another biological occurrence that creates a complex pattern, biovermiculation growth, has also been shown to be simulated by CA. Figure 2.2 [8] shows a CA simulation of biovermiculation growth. They are found within cave walls and are caused by bacteria, slime, clay and other minerals. As mentioned in [2] these patterns are of interest because similar lifeforms may be able to live within the caves of Mars.

2.4.3 *Connections with Other Models*

Wolfram also points out an interesting connection between CA and partial differential equations [15]. Wolfram tried to create a model for a continuous CA, that is, a CA with no cells but an infinite amount of continuous points. Also the values of each point are an amount of gray ranging from completely filled in, or black, to empty,

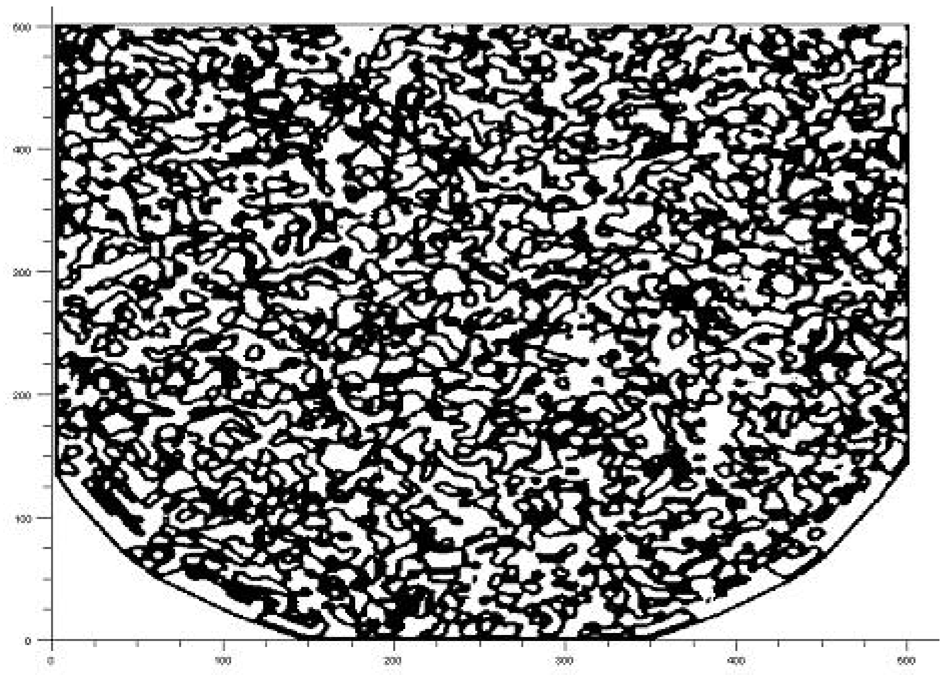

Fig. 2.2: Cellular automata simulation of biovermiculation growth on a cave wall after many iterations.

or white. Initially, it would seem difficult to develop rules for a continuous system. Wolfram however shows that partial differential equations themselves could be used as the rules for a continuous CA. The difference between what Wolfram has shown and my proposed thesis is that I want to stay in the discrete realm. I want to find out is there a way to transform the differential equations into a discrete form of rules.

CAs are also of direct importance to computer science. Wolfram shows implicitly that one dimensional CAs with the right setup are equivalent to Turing machines [15]. The type of CA used is a mobile automata, which has a place marker, and the only cell that has the rules applied to it each time period is the marked cell. Each cell in the one dimensional CA is either filled or not, representing a tape of binary digits. Included in the state is also an arrow which serves as the marker. This arrow represents the head of the Turing machine. For each transition the marker can either move to the left or the right of the current cell. The arrow can also point either left, right, or down to indicate additional states rather than just the cell being filled in or empty.

2.4.4 Other Differential Equation to Cellular Automata Translation Attempts

Ever since Wolfram linked CAs and differential equations over a decade ago [14], others have also tried to detail that link. A generic method for converting differential equations was discovered by researchers at Beijing Polytechnic University [4], which include:

1. Convert a differential equation in the form of $\frac{dx}{dt} = f(x, y)$ to a finite difference equation $x(t + 1) = x(t) + f(x(t), y(t))$.
2. Replace all variables like x and y with discrete state variables used by the CA.
3. The transition function then becomes 1+f.

These steps were applied to differential equations that describe tumor growth. Some of these steps are generic and I will attempt to produce a method in my thesis that is more detailed, whether it be based upon these steps or a completely different process.

3. DIFFERENTIAL EQUATION CONVERSION

In this chapter I will first describe the general characteristics of biological differential equations that can be simulated by cellular automata (CA). This is found in Section 3.1 on page 17. Next I will present a survey of biological equations that fit these characteristics in Section 3.2 on page 18, some of which are explained in more detail in Section 2.2 on page 6. The following section, Section 3.3 on page 21, will summarize the biological differential equations into a couple generalized forms. Then in Section 3.4 on page 21, I will show how both Forward and Backward Euler's methods can be used to discretize the general differential equation forms so that it can be used as rules for CA.

3.1 Differential Equation Characteristics

The class of differential equations that would be useful to simulate are those that contain both time and spatial parameters, explicitly the partial differential equations with respect to time that contain within their definitions gradient or Laplacian terms with respect to space. Simply put, these equations explain how biological values change with time depending upon how their spacial neighbors are changing. This is also a rough definition of how CA rules work because they explain what is happening within a particular cell as time changes according to the values of their neighbors.

For this document the general name of the biological functions that have these characteristics will be $u(t, x)$ where t is the position in time for the spacial position x. I will be using subscripts to indicate different positions in time and space so $u(t_i, x_j)$ is equal to the value of a vector at position j at time i. For the function u to refer to higher dimensions, extra position parameters can be tacked on for each dimension (Ex: $u(t_i, x_j, y_k, z_l)$). To make some of the following equations easier to read I will let $u_{i,j} = u(t_i, x_j)$.

The function $f(t, x)$ will be the name of the differential equation of $u(t, x)$ with respect to time. In other words $f(x, t) = \frac{\partial u}{\partial t}$. Normally the function $f(t, x)$ will not actually have the terms x and t within the function but instead are passed in some form of $u_{i,j}$. Because of this I will simply use u as the parameter itself: $f(u_{i,j})$. The following section presents examples of biological equations that fit these characteristics.

3.2 Biological Equations

This section contains a small survey of spatial biological equations in order to determine some general equation forms to be used.

3.2.1 Fick's Law

Fick's law is a differential equation for population density [10]:

$$\frac{\partial P}{\partial t} = f(t, x, P) + d\nabla_x^2 P \tag{3.1}$$

Here P represents population density. The equation says that populations in more dense areas will move to less dense areas, depicted by $d\nabla_x^2 P$ where d is a diffusion constant. The function $f(t, x, P)$ represents the reaction rate, where the size of the population is changed due to non-density factors like birth or death.

In the most general case, $f(t, x, P) = kP$, which gives:

$$\frac{\partial P}{\partial t} = kP + d\nabla_x^2 P \tag{3.2}$$

3.2.2 Random Walk

The number of random walker entities at a specific location is described by [10]:

$$\frac{\partial P}{\partial t} = D\frac{\partial^2 P}{\partial t^2} - V\frac{\partial P}{\partial x} \tag{3.3}$$

Here D is another diffusion constant of the movement in $D\frac{\partial^2 P}{\partial t^2}$. The second term $V\frac{\partial P}{\partial x}$ allows for a bias within the random walk.

3.2.3 Predator-Prey

The following equations can be used to simulate predator and prey population using a variation of the Lotka-Voltera equations "with self-limitation of the prey and Holling II functional response" [7]:

$$\frac{\partial r_i}{\partial t} = \left(a - br - \frac{p}{1+r}\right) r_i + d_{r_i}\nabla^2 r_i \tag{3.4}$$

$$\frac{\partial p_i}{\partial t} = \left(\frac{r}{1+r} - c\right) p_i + d_{p_i}\nabla^2 p_i \tag{3.5}$$

Here r_i and p_i respectively refer to the predator and prey population i. Without the subscript i, r and p refer to the total sum of the population for predators and prey.

3.2.4 Another Predator-Prey Model

The following is another predator-prey relationship where random fluctuations are of the type white noise, allowing the Fokker-Planck equation to be used[1]:

$$\frac{\partial P}{\partial t} = \frac{\partial}{\partial x}[a(x)P] + \frac{1}{2}\frac{\partial^2}{\partial x^2}[b(x)P] \tag{3.6}$$

where

$$a(x) = kx\frac{1-\left(\frac{x}{\theta}\right)^{\alpha}}{\alpha} + \sigma^2\frac{x}{2} \tag{3.7}$$

$$b(x) = \sigma^2 x^2 \tag{3.8}$$

3.2.5 Desert Vegetation Patterns

These are equations that describe plant growth through biomass density n(x,t), and water density w(x,t) as describes in detail in Section 2.2.2 [13]:

$$\frac{\partial n}{\partial t} = \frac{yw}{1+\sigma w}n - n^2 - \mu n + \nabla^2 n \tag{3.9}$$

$$\frac{\partial w}{\partial t} = p - (1-\rho n)w - w^2 n + \delta\nabla^2(w-\beta n) - v\frac{\partial(w-\alpha n)}{\partial x} \tag{3.10}$$

3.2.6 Morphogenesis

Here X and Y represent the amounts of 2 morphogens within a ring of cells. A subscript r indicates the amount X or Y corresponds to the particular cell r. This results in the following two formulas[12]:

$$\frac{dX_r}{dt} = f(X_r, Y_r) + \mu(X_{r+1} - 2X_r + X_{r-1}) \tag{3.11}$$

$$\frac{dY_r}{dt} = g(X_r, Y_r) + \nu(Y_{r+1} - 2Y_r + Y_{r-1}) \tag{3.12}$$

If the ring is thought of as continuous tissue, the equations become similar to ones shown in previous subsections:

$$\frac{\partial X}{\partial t} = a(X-h) + b(Y-k) + \frac{\mu'}{\rho_2}\frac{\partial^2 X}{\partial \theta^2} \tag{3.13}$$

$$\frac{\partial Y}{\partial t} = c(X-h) + d(Y-k) + \frac{\nu'}{\rho_2}\frac{\partial^2 X}{\partial \theta^2} \tag{3.14}$$

Where θ describes the position of a cell by an angle within the ring of cells.

3.3 Differential Equation General Form

The following form best encapsulates the equations 3.2, 3.3, 3.4, 3.5, 3.9 and 3.10 from Section 3.2:

$$f(u_{i,j}) = \frac{\partial u}{\partial t} = m(u_{i,j}) + \nabla_x^2 n(u_{i,j}) + \nabla_x o(u_{i,j}) \tag{3.15}$$

In this form all of the terms that only contain $u_{i,j}$ are contained in $m(u_{i,j})$. All of the terms that have the Laplacian with respect to space applied to them are contained in $n(u_{i,j})$. The term $o(u_{i,j})$ contains the elements within the formula that have the gradient applied them.

3.4 Discretizing Partial Differential Equations

3.4.1 Forward Euler's

A common and simple way to solve an ordinary differential equation by approximation is by using Forward Euler's Method, which is the following formula:

$$u_{i+1,j} = u_{i,j} + h_t f(u_{i,j}) \tag{3.16}$$

h_t is defined as the interval between each t_i value. $f(u_{i,j})$ is the differential equation of $u_{i,j}$. I will substitute Equation 3.15 for $f(u_{i,j})$ but first I must discretize the equation by the space component, removing the gradient and Laplacian from the equation. The terms $\nabla_x^2 n(u_{i,j})$ and $\nabla_x o(u_{i,j})$ can be substituted using the 3 point formula (shown in Section 2.3):

$$\nabla_x^2 n(u_{i,j}) \approx \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} \tag{3.17}$$

$$\nabla_x o(u_{i,j}) \approx \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \tag{3.18}$$

Substituting these into equation 3.15 $f(u_i)$ gives:

$$f(u_{i,j}) = m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \tag{3.19}$$

If I substitute $f(u_{i,j})$ into the Forward Euler equation I get:

$$u_{i+1,j} = u_{i,j} + h_t \left(m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right) \tag{3.20}$$

From this equation I can construct a cellular automata model. If I let u be the cells within the cellular automata, then Equation 3.20 can be used as the simple rule to change u from one time period to another. As with other CA models, the equation includes the nearest neighbors of a particular cell within the CA rule. This model is mathematically much simpler than the differential equation (Eq. 3.3) because the rule only includes the operations addition, subtraction, multiplication, and division.

3.4.2 Solving Backward Euler's

Forward Euler's method is nice in that it is explicit and gives a direct equation, but it can break down quickly as h_t becomes large. The Backward Euler's method, however, is more stable as h_t becomes large. It has a subtle difference from Forward Euler's in that $u_{i+1,j}$ is used within the formula itself:

$$u_{i+1,j} = u_{i,j} + h_t f(u_{i+1,j}) \tag{3.21}$$

I do not know what $u_{i+1,j}$ is equal to, but I can solve the last part of the equation in terms of $u_{i,j}$ [5]. First I subtract $u_{i,j}$ from both sides of equation 3.21:

$$\Delta u = h_t f(u_{i,j} + \Delta u) \tag{3.22}$$

Now I can change the right hand side of equation 3.21 with its first order Taylor series. Then I solve for Δu:

$$\begin{aligned}
\Delta u &= h_t f(u) + h_t \frac{\partial f}{\partial u_{i,j}} \Delta u && (3.23)\\
\Delta u - h_t \frac{\partial f}{\partial u} \Delta u &= h_t f(u_{i,j}) && (3.24)\\
(1 - h_t \frac{\partial f}{\partial u}) \Delta u &= h_t f(u_{i,j}) && (3.25)\\
\Delta u &= \frac{h_t f(u_{i,j})}{1 - h_t \frac{\partial f}{\partial u}} && (3.26)
\end{aligned}$$

Now I add $u_{i,j}$ back into equation 3.26 to get back the Backward Euler's equation in terms of only $u_{i,j}$:

$$u_{i+1,j} = u_{i,j} + \frac{h_t f(u_{i,j})}{1 - h_t \left.\frac{\partial f(u)}{\partial u}\right|_{i,j}} \tag{3.27}$$

Now all I have to do is substitute in $f(u_{i,j})$ from equation 3.19:

$$\begin{aligned}
u_{i+1,j} &= u_{i,j} + \frac{h_t \left(m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right)}{1 - h_t \frac{\partial}{\partial u} \left(m(u) + \frac{n(u) - 2n(u) + n(u)}{h_x^2} + \frac{o(u) - o(u)}{2h_x} \right) \Big|_{i,j}} && (3.28)\\
&= u_{i,j} + && (3.29)\\
&\frac{h_t \left(m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right)}{1 - h_t \left(\left.\frac{\partial m(u)}{\partial u}\right|_{i,j} + \frac{\left.\frac{\partial n(u)}{\partial u}\right|_{i,j+1} - 2 \left.\frac{\partial n(u)}{\partial u}\right|_{i,j} + \left.\frac{\partial n(u)}{\partial u}\right|_{i,j-1}}{h_x^2} + \frac{\left.\frac{\partial o(u)}{\partial u}\right|_{i,j+1} - \left.\frac{\partial o(u)}{\partial u}\right|_{i,j-1}}{2h_x} \right)}
\end{aligned}$$

Although this formula may seem complicated, it reduces greatly if both $n(u)$ and $o(u)$ contain u's with a degree less than two. In this case, the two fractions in the denominator containing $n(u)$ and $o(u)$ become zero when I take the partial derivative. This gives the equation:

$$u_{i+1,j} = u_{i,j} + \frac{h_t \left(m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right)}{1 - h_t \frac{\partial m(u)}{\partial u}\Big|_{i,j}} \tag{3.30}$$

This equation can also be used as a rule of a CA for u cells. This equation does include partial differentiation, but once $m(u)$, $n(u)$, and $o(u)$ have been substituted into the formula, the partial derivative can be evaluated, giving a formula with only addition, subtraction, multiplication, and division.

4. STABILITY AND Z-TRANSFORM OF THE GENERAL LINEAR FORM

In this chapter, I discuss the stability of the Euler functions found in the previous chapter. In Section 4.1 on page 26 I present a general form of the biological equations that only include linear terms of u that I refer to in the rest of the document as the general linear form. In subsection 4.1.1, I use a Z-transform to find out what values are stable for the general linear equation form using Forward Euler's method. Then in Subsection 4.1.2, I use the same process on the Backward Euler's equation I created.

In both Subsections 4.1.1 and 4.1.2, I find the stability of equations by first manipulating the formula so that the u variables can be transformed easily. Next I preform the Z-transform and solve for U_j, the u variable after it has gone through the Z-transform. With the resulting formula I find the poles and zeros for the z variable. If given some function that looks like $\frac{f(z)}{g(z)}$, the zeros of the function are the z values where $f(z) = 0$ and the poles of the function are the values of z that make $g(z) = 0$. The poles of the function describe the Region Of Convergence, the area where z exists. If the Region of Convergence contains the unit circle on the complex plane, then the equation will be stable. Therefore, by setting the poles to less than one (the radius of the unit circle), the inequalities will become constraints on stability.

In Section 4.2 on page 31, I will then compare the resulting poles and zeros from Subsections 4.1.1 and 4.1.2 to show that the modified Backward Euler's (Eq. 3.30) is theoretical more stable than the modified Forward Euler's (Eq. 3.20), although this does not hold true in practice as shown in the next chapter.

4.1 General Linear Equation Stability

For the remainder of this thesis I will be focusing on a particular subset of the general partial differential equation form, one that uses linear terms with respect to u. I will use the following equation to represent the general linear form and parse it into the format of Equation 3.15:

$$f(u) = a_1 u + b_1 + \nabla_x^2(a_2 u) + \nabla_x(a_3 u) \tag{4.1}$$

I can assign parts of the equations to the following functions:

$$m(u) = a_1 u + b_1 \tag{4.2}$$

$$n(u) = a_2 u \tag{4.3}$$

$$o(u) = a_3 u \tag{4.4}$$

Now I plug these substitutions into the final Forward and Backward Euler's equations and compare their stability.

4.1.1 Linear Forward Euler's Equation Stability

If I substitute equations 4.2, 4.3, and 4.4 into equation 3.20 I get:

$$\begin{aligned} u_{i+1,j} = {} & u_{i,j} + h_t \Bigg(a_1 u_{i,j} + b_1 + \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \\ & \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x} \Bigg) \end{aligned} \tag{4.5}$$

Now I will multiply by the necessary values so that $2h_x^2$ is the common denominator for the fractions:

$$\begin{aligned} u_{i+1,j} &= u_{i,j} + h_t\left(\frac{2a_1h_x^2u_{i,j}}{2h_x^2} + \frac{2b_1h_x^2}{2h_x^2} + \right. \\ &\left. \frac{(2)(a_2u_{i,j+1} - 2a_2u_{i,j} + a_2u_{i,j-1})}{2h_x^2} + \frac{(h_x)(a_3u_{i,j+1} - a_3u_{i,j-1})}{2h_x^2}\right) \end{aligned} \tag{4.6}$$

Then I bring out a $\frac{1}{2h_x^2}$ and multiply the left term by $\frac{2h_x^2}{2h_x^2}$ to get:

$$\begin{aligned} u_{i+1,j} &= \frac{2h_x^2u_{i,j}}{2h_x^2} + \frac{h_t}{2h_x^2}\left(2a_1h_x^2u_{i,j} + 2b_1h_x^2 + \right. \\ &\left. (2)(a_2u_{i,j+1} - 2a_2u_{i,j} + a_2u_{i,j-1}) + (h_x)(a_3u_{i,j+1} - a_3u_{i,j-1})\right) \qquad (4.7) \\ &= \frac{h_t}{2h_x^2}\left(\frac{2h_x^2u_{i,j}}{h_t} + 2a_1h_x^2u_{i,j} + 2h_x^2b_1 + \right. \\ &\left. (2)(a_2u_{i,j+1} - 2a_2u_{i,j} + a_2u_{i,j-1}) + (h_x)(a_3u_{i,j+1} - a_3u_{i,j-1})\right) \qquad (4.8) \\ &= \frac{h_t}{2h_x^2}\left(\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(u_{i,j}) + (2a_2 - a_3h_x)(u_{i,j-1}) + \right. \\ &\left. (2a_2 + a_3h_x)(u_{i,j+1}) + 2b_1h_x^2\right) \qquad (4.9) \end{aligned}$$

Now I take the Z-Transform and solve for U_j to get the following equation, eliminating the i index:

$$\begin{aligned} U_j = \frac{h_t}{2h_x^2}&\left(\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1}U_j) + (2a_2 - a_3h_x)(z^{-1}U_{j-1}) + \right. \\ &\left. (2a_2 + a_3h_x)(z^{-1}U_{j+1}) + 2b_1h_x^2\right) \end{aligned} \tag{4.10}$$

I will gather all of the U_j terms on the left side of the equation:

$$\begin{aligned} \left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)U_j = \frac{h_t}{2h_x^2}((2a_2 - a_3h_x)(z^{-1}U_{j-1}) + \\ (2a_2 + a_3h_x)(z^{-1}U_{j+1}) + 2b_1h_x^2) \end{aligned} \tag{4.11}$$

Then I divide by the coefficients of U_j to solve for U_j:

$$U_j = \frac{\frac{h_t}{2h_x^2}((2a_2 - a_3h_x)(z^{-1}U_{j-1}) + (2a_2 + a_3h_x)(z^{-1}U_{j+1}) + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{4.12}$$

Now I need to find the poles and zeros of the resulting equation. First I will change the z^{-1} to $\frac{1}{z}$, and then multiply the entire fraction by by $\frac{z}{z}$:

$$U_j = \frac{\frac{h_t}{2h_x^2}\left((2a_2 - a_3h_x)(U_{j-1})\frac{1}{z} + (2a_2 + a_3h_x)(U_{j+1})\frac{1}{z} + 2b_1h_x^2\right)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)\frac{1}{z}\right)} \tag{4.13}$$

$$= \frac{\frac{h_t}{2h_x^2}((2a_2 - a_3h_x)(U_{j-1}) + (2a_2 + a_3h_x)(U_{j+1}) + 2b_1h_x^2z)}{\left(z - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)\right)} \tag{4.14}$$

The equation will remain stable as long as the absolute value of the poles and zeros are less than one. To find the zeros, I set the numerator of Equation 4.14 equal to zero and solve for z. The resulting zero value for z must be less than 1 in order for the equation to be stable, which gives the following condition for stability:

$$1 > \left|\frac{-1}{2b_1h_x^2}((2a_2 - a_3h_x)(U_{j-1}) + (2a_2 + a_3h_x)(U_{j+1}))\right| \tag{4.15}$$

Unfortunately, this constraint contains U_{j-1} and U_{j+1} which change with each time step, and therefore, they are not constant. In Appendix A on page 67, I will substitute these values with constant approximations to see if the new constraints help to further describe the stability of the Linear Forward Euler equation.

Next I need to find the poles of Equation 4.14 by setting the denominator equal to zero and solving for z. That z value must also be less than one for the equation to be stable, which gives the following condition for stability:

$$1 > \left|\frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)\right| \tag{4.16}$$

$$> \left|1 + a_1h_t - \frac{2a_2h_t}{h_x^2}\right| \tag{4.17}$$

4.1.2 Linear Backward Euler's Equation Stability

Now I do the same operations on the Backward Euler's formula, Equation 3.30, by first substituting in $m(u)$, $n(u)$, and $o(u)$:

$$\begin{aligned} u_{i+1,j} &= u_{i,j} + \frac{h_t}{1 - h_t \frac{\partial(a_1 u + b_1)}{\partial u}\Big|_{i,j}} \Big(a_1 u_{i,j} + b_1 + \\ &\quad \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \\ &\quad \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x}\Big) \qquad (4.18) \\ &= u_{i,j} + \frac{h_t}{1 - a_1 h_t} \Big(a_1 u_{i,j} + b_1 + \\ &\quad \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \\ &\quad \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x}\Big) \qquad (4.19) \end{aligned}$$

Again I will make $2h_x^2$ the common denominator, and then I bring out a $\frac{1}{2h_x^2}$ and multiply the left term by $\frac{2h_x^2}{2h_x^2}$ to get:

$$\begin{aligned} u_{i+1,j} &= \frac{2h_x^2 u_{i,j}}{2h_x^2} + \frac{h_t}{2h_x^2(1 - h_t a_1)}(2a_1 h_x^2 u_{i,j} + 2b_1 h_x^2 + \\ &\quad (2h_x)(a_2 u_{i,j+1} + -2a_2 u_{i,j} + a_2 u_{i,j-1}) + (h_x^2)(a_3 u_{i,j+1} - a_3 u_{i,j-1})) \qquad (4.20) \\ &= \frac{2h_x^2(1 - a_1 h_t) u_{i,j}}{2h_x^2(1 - a_1 h_t)} + \frac{h_t}{2h_x^2(1 - a_1 h_t)}(2a_1 h_x^2 u_{i,j} + 2b_1 h_x^2 + \\ &\quad (a_2 u_{i,j+1} + -2a_2 u_{i,j} + a_2 u_{i,j-1})(2h_x) + (a_3 u_{i,j+1} - a_3 u_{i,j-1})(h_x^2)) \qquad (4.21) \\ &= \frac{h_t}{2h_x^2(1 - a_1 h_t)} \left(\frac{2h_x^2(1 - a_1 h_t) u_{i,j}}{ht} + 2a_1 h_x^2 u_{i,j} + 2b_1 h_x^2 + \right. \\ &\quad \left. (a_2 u_{i,j+1} + -2a_2 u_{i,j} + a_2 u_{i,j-1})(2h_x) + (a_3 u_{i,j+1} - a_3 u_{i,j-1})(h_x^2)\right) \qquad (4.22) \\ &= \frac{h_t}{2h_x^2(1 - a_1 h_t)} \left(\left(\frac{2h_x^2(1 - a_1 h_t)}{h_t} + 2a_1 h_x^2 - 4a_2\right)(u_{i,j}) + (2a_2 - a_3 h_x)(u_{i,j-1}) + \right. \\ &\quad \left. (2a_2 + a_3 h_x)(u_{i,j+1}) + 2b_1 h_x^2\right) \qquad (4.23) \end{aligned}$$

Now I will take the Z-Transform and solve for U_j, again removing the index i from the equation:

$$\begin{aligned} U_j &= \frac{h_t}{2h_x^2(1-a_1h_t)}\left(\left(\frac{2h_x^2(1-a_1h_t)}{h_t}+2a_1h_x^2-4a_2\right)(z^{-1}U_j)+\right. \\ &\quad \left.(2a_2-a_3h_x)(z^{-1}U_{j-1})+(2a_2+a_3h_x)(z^{-1}U_{j+1})+2b_1h_x^2\right) \end{aligned} \tag{4.24}$$

I will gather all of the U_j terms on the left side of the equation:

$$\begin{aligned} &\left(1-\frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2(1-a_1h_t)}{h_t}\right.\right. \\ &\qquad \left.\left.+2a_1h_x^2-4a_2\right)(z^{-1})\right)U_j = \frac{h_t}{2h_x^2(1-h_ta_1)}((2a_2-a_3h_x)(z^{-1}U_{j-1})+ \\ &\qquad (2a_2+a_3h_x)(z^{-1}U_{j+1})+2b_1h_x^2) \end{aligned} \tag{4.25}$$

Then I divide by the coefficients of U_j to solve for U_j:

$$U_j = \frac{\frac{h_t}{2h_x^2(1-a_1h_t)}((2a_2-a_3h_x)(z^{-1}U_{j-1})+(2a_2+a_3h_x)(z^{-1}U_{j+1})+2b_1h_x^2)}{\left(1-\frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2(1-a_1h_t)}{h_t}+2a_1h_x^2-4a_2\right)(z^{-1})\right)} \tag{4.26}$$

I then multiply by $\frac{z}{z}$ and find the poles and zeros:

$$U_j = \frac{\frac{h_t}{2h_x^2(1-a_1h_t)}((2a_2-a_3h_x)(U_{j-1})+(2a_2+a_3h_x)(U_{j+1})+2b_1h_x^2z)}{\left(z-\frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2(1-a_1h_t)}{h_t}+2a_1h_x^2-4a_2\right)\right)} \tag{4.27}$$

Again, I can find the zeros by setting the numerator of Equation 4.27 equal to zero and solving for z. The resulting zero value for z must be less than one in order for the equation to be stable, which gives the following condition for stability for the Backward Euler's equation:

$$1 > \left|\frac{-1}{2b_1h_x^2}((2a_2-a_3h_x)(U_{j-1})+(2a_2+a_3h_x)(U_{j+1}))\right| \tag{4.28}$$

Again I run into the problem of the nonconstant U_{j-1} and U_{j+1} values. As mentioned before, I attempt to address this issue in Appendix A on page 67.

Next I find the poles of Equation 4.27 to find the second stability condition by setting the denominator equal to zero and solving for z. That z value must also be less than one for the equation to be stable, which gives the following condition for stability:

$$1 > \left|\frac{h_t}{1-a_1h_t}\left(\frac{1-a_1h_t}{ht}+a_1-\frac{2a_2}{h_x^2}\right)\right| \tag{4.29}$$

$$> \left|1+\frac{a_1h_t}{1-a_1h_t}-\frac{2a_2h_t}{(1-a_1h_t)h_x^2}\right| \tag{4.30}$$

4.2 General Linear Equation Comparative Stability Analysis

Now I compare the stability constraints found for both the Backward and Forward Euler's functions.

For the constraint created by the zeros, both formulas had the same constraint (Equations 4.15 and 4.28):

$$1 > \left|\frac{-1}{2b_1h_x^2}((2a_2-a_3h_x)(U_{j-1})+(2a_2+a_3h_x)(U_{j+1}))\right| \tag{4.31}$$

In this equation, h_t does not appear to affect its stability at all. This is incorrect, however, because the h_t value is a part of the U_{j-1} and U_{j+1}, so this effect is hidden. As for h_x, if h_x does become large, the h_x^2 in the denominator will overwhelm everything else and make the entire right side of the formula small, keeping it stable.

For the constraints created by the poles, the Forward Euler constraint from Equation 4.17 is:

$$1 > \left|1+a_1h_t-\frac{2a_2h_t}{h_x^2}\right| \tag{4.32}$$

and the constraint from the Backward Euler poles (Equation 4.30) is:

$$1 > \left|1+\frac{a_1h_t}{1-a_1h_t}-\frac{2a_2h_t}{(1-a_1h_t)h_x^2}\right| \tag{4.33}$$

These two formulas show that there needs to be a balancing of h_t and $\frac{h_t}{h_x^2}$ in order for the formula to remain stable. In both formulas the second term contains the h_t term and if it becomes large it can make the whole formula become larger than 1. The third term, however, contains a $\frac{h_t}{h_x^2}$ term, which means that as h_x shrinks smaller, the term becomes larger. The third term is negative and as a result if h_t becomes large and h_x^2 becomes smaller at the same rate, the second and third terms will cancel each other out. The outcome will be that the overall formula is smaller than 1.

The difference between these two formulas show why the Backward Euler's formula in general will be more stable than the Forward Euler's formula as h_t gets large. For the second constraint, both terms are divided by $1 - a_1 h_t$ so that as h_t becomes large, the terms will be divided by a larger denominator, creating a smaller result and allowing simulations to be more stable as they last longer. If h_t is small, then there is little difference between the two Euler's formulas because the two terms in Backward Euler's will be divided by a number close to 1. Therefore, I would recommend from this information to use the Backward Euler's formula, because it is more stable when h_t is big and there is no cost to stability when h_t is small. However, as will be shown in Section B.2 on page 90, $a_1 h_t$ at maximum is 0.1, meaning both formulas are nearly the same even if h_t becomes large.

5. CONVERGENCE MAPS AND CELLULAR AUTOMATA SIMULATION RESULTS

In this chapter I analyze convergence maps, graphs that show the areas of convergence for my CA models, based upon how space and time are discretized (the sizes of h_x and h_t). In Section 5.1, on page 34, I define how the CA simulations are run to create the convergence maps, and I make some observations about the maps. Then in Section 5.2, on page 38, I graph the theoretical stability constraints from Section 4.2 and show they form the boundaries of convergence. I also identify a third boundary not explained by the theoretical stability constraints that I refer to as the a_3 vertical boundary because it draws a distinct nearly vertical line in the maps between convergence and divergence and because its position is based upon the a_3 parameter. In Section 5.3, on page 46, I calculate the error between the theoretical stability boundaries and the actual convergence map boundaries. In Section 5.4, on page 49, I show that the speed of the CA simulation is faster when the discretization parameters are closer to the lower convergence boundary. Finally in Section 5.5, on page 50, I use this information to propose a set of guidelines to help pick h_x and h_t.

5.1 Convergence Maps

5.1.1 Convergence Map Construction and Cellular Automata Simulation Methodology

In this section there are several graphs depicting convergence maps, normally using h_t and h_x as parameters. The maps were constructed by running either the Forward or Backward Euler's functions (Eqs. 4.5 and 4.19) within Scilab for the general linear form, until one of three conditions were met. The first condition was that the values converged, which for my tests meant that $\left|\left|u_{i+1} - u_i\right|\right|_2 < 10^{-10}$. The second was that u was going to diverge, through the condition $\left|\left|u_{i+1} - u_i\right|\right|_2 > 10^{10}$. The final condition was that the Euler's formula being tested went through four thousand iterations without meeting either the first or second conditions. In the convergence maps that follow, if the Scilab program ended due to convergence, a blue dot is placed on the map for corresponding h_t and h_x values used in the program. Similarly, green was used for h_t and h_x values that diverged, and red was used for those values that neither converged or diverged in the maximum number of iterations allotted.

The maps were created by running the Euler's functions using $u_0 = [1\ 2\ 3\ 4\ 5]$. The left and right boundary cells ($u_{i,0}$ and $u_{i,6}$) were set as value zero and do not change from one time period to another. The Scilab code used to run the CA simulations are found in Section C.1 on page 98. The Scilab code to plot the simulation data is in Section C.2 on page 104.

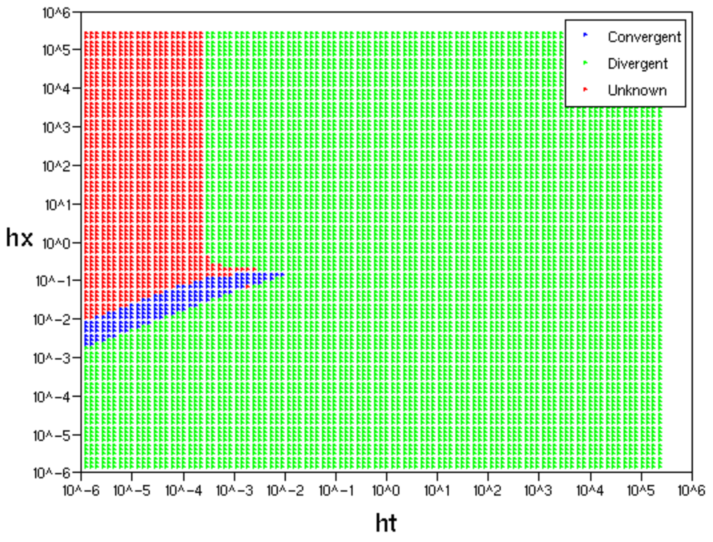

Fig. 5.1: Convergence map composed of 10,000 simulations with varying h_t and h_x values for the Forward Euler's function.

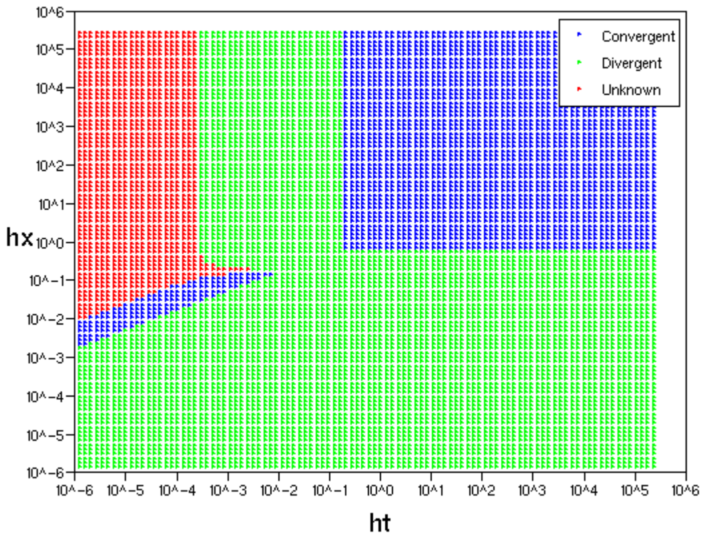

Fig. 5.2: Convergence map composed of 10,000 simulations with varying h_t and h_x values for the Backward Euler's function.

5.1.2 Function Convergence Analysis and Convergence Values

The first convergence map in Figure 5.1 is of the Forward Euler's Linear equation with a_1=10, a_2=1, a_3=1, and b_1=1, to give a basic idea of how the parameters h_t and h_x affect the convergence. Figure 5.2 shows a convergence map of the Backward Euler's function with the same parameters. The obvious difference being the upper right quadrant of the Backward Euler's converges while the Forward Euler's does not. I will analyze the Euler's functions by dividing the convergence maps into four major quadrants.

The first quadrant is the upper right corner where both h_t and h_x become large. The values of u do not converge. In the linear Forward Euler's equation (Eq. 4.5), the last two terms go to zero, because they are divided by a form of h_x, which becomes large. This leaves the equation: $u_{i+1,j} = u_{i,j} + h_t\,(a_1 u_{i,j} + b_1)$. With a_1 being positive, $h_t b_1$ will be continually added, which is the likely cause for this quadrant to diverge.

The situation is different for the Backward Euler's equation for this quadrant, because the entire upper right quadrant will converge. The size of h_x creates a similar situation to the Forward Euler's equation, making the right most two terms go to zero. The difference in the Backward Euler's formula is that the h_t term becomes $\frac{h_t}{1-a_1 h_t}$. Because h_t is also large, this term effectively becomes 1. This leaves the equation: $u_{i+1,j} = u_{i,j} + a_1 u_{i,j} + b_1$. Simulations have shown that $u_{i,j}$ will converge at the value $-\frac{b_1}{a_1}$ for the entire quadrant. This is possibly just an artifact of using Backward Euler's.

For the second quadrant, where h_t is small and h_x is large, both the Backward and Forward Euler's equations will diverge if a_1 is positive. With h_t being close to zero, it is a similar situation to the first quadrant, except that it takes a much longer time to diverge. If a_1 is negative however, $u_{i,j}$ will converge to $-\frac{b_1}{a_1}$. A convergence map with a_1 as negative is shown in Figure 5.3.

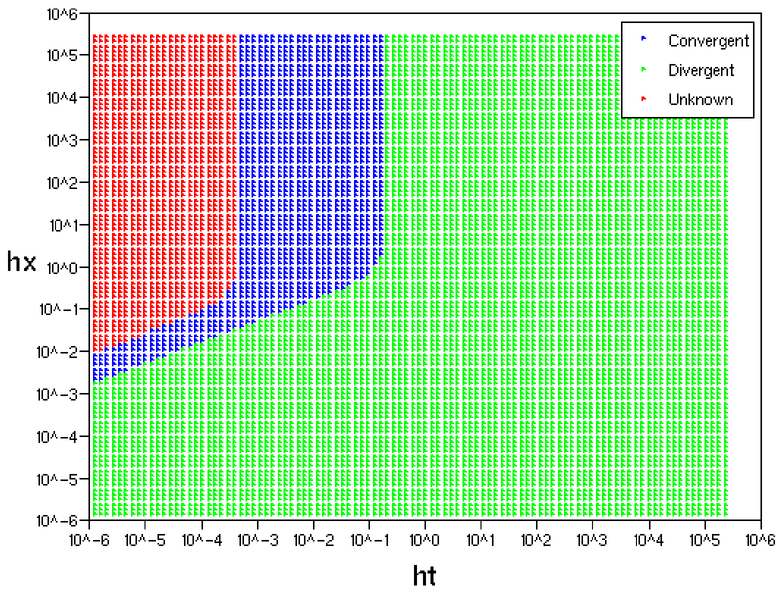

Fig. 5.3: Convergence map composed of simulations with varying h_t and h_x values for the Forward Euler's function when a_1 is negative.

The third quadrant, where both h_t and h_x are small, is split into two halves on both Backward and Forward Euler's equations. The upper left half of this quadrant converges while the lower right does not. The half that does converge represents the balance necessary between h_t and h_x as mentioned within Section 4.2. The line separating the two halves between convergence and divergence has a slope of ch_x^2 for some constant c. The value of c is examined in Section B.2 on page 90 in Table B.1.

For the rest of this thesis I will be focused upon the third quadrant area of convergence. This is due to the values that are converged upon in the area appear to be of importance. The convergence values are nearly the same, given the same h_x value, even with different h_t values. In other words the manner in which space is discretized affects the outcome of the convergence values but how time is discretized does not.

The rest of the third quadrant and the entire fourth quadrant in the bottom right of the map do not converge, most likely for the same reason. As h_t becomes big and h_x becomes small, the last two terms divided by h_x become very large. Because h_t is also large, it cannot be used to slow down the growth of those last two factors. Also because the terms are linear and u_0's values are very close to each other, the last two terms are near zero in the numerator and therefore are not that big. However, the last cell of u ($u_{i,5}$) is next to the boundary cell ($u_{i,6}$), which always has the value zero, which means the terms no longer cancel each other out in the second to last term: $\frac{a_2u_{i,j+1}-2a_2u_{i,j}+a_2u_{i,j-1}}{h_x^2}$. In other words, for the last value of u, the quantities $a_2u_{i,j-1}$ and $a_2u_{i,j}$ are nearly the same and cancel out, but $a_2u_{i,j+1} = 0$, leaving $\frac{-a_2u_{i,j}}{h_x^2}$. This term then grows exponential because h_x^2 is very small. The absolute value of the last term in u will grow towards infinity in these areas.

5.2 Convergence Boundaries

Now I will find the theoretical boundaries of the convergence map by taking the inequalities in Equations 4.32 and 4.33 and setting them equal to one. By solving these equations for h_x in terms of h_t I can graph this boundary on top of our convergence maps.

5.2.1 Forward Euler's Convergence Boundaries

First, for the Forward Euler's poles equation I begin with:

$$1 = \left|1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2}\right| \tag{5.1}$$

In order to eliminate the absolute value, I will solve for two different equations, first assuming the right side is positive and second that it is negative. First I assume the right side is positive to obtain the first constraint:

$$\begin{aligned}
1 &= 1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2} && (5.2)\\
a_1 h_t &= \frac{2a_2 h_t}{h_x^2} && (5.3)\\
h_x^2 &= \frac{2a_2 h_t}{a_1 h_t} && (5.4)\\
h_x &= \sqrt{\frac{2a_2 h_t}{a_1 h_t}} && (5.5)\\
&= \sqrt{\frac{2a_2}{a_1}} && (5.6)
\end{aligned}$$

The second constraint assumes the right side is negative:

$$\begin{aligned}
1 &= -1 - a_1 h_t + \frac{2a_2 h_t}{h_x^2} && (5.7)\\
2 + a_1 h_t &= \frac{2a_2 h_t}{h_x^2} && (5.8)\\
h_x^2 &= \frac{2a_2 h_t}{2 + a_1 h_t} && (5.9)\\
h_x &= \sqrt{\frac{2a_2 h_t}{2 + a_1 h_t}} && (5.10)
\end{aligned}$$

This gives the following two boundary equations for Forward Euler's:

$$h_x = \sqrt{\frac{2a_2}{a_1}} \tag{5.11}$$

and

$$h_x = \sqrt{\frac{2a_2 h_t}{2 + a_1 h_t}} \tag{5.12}$$

Figure 5.4 is a convergence map of a Forward Euler's function with these two constraints.

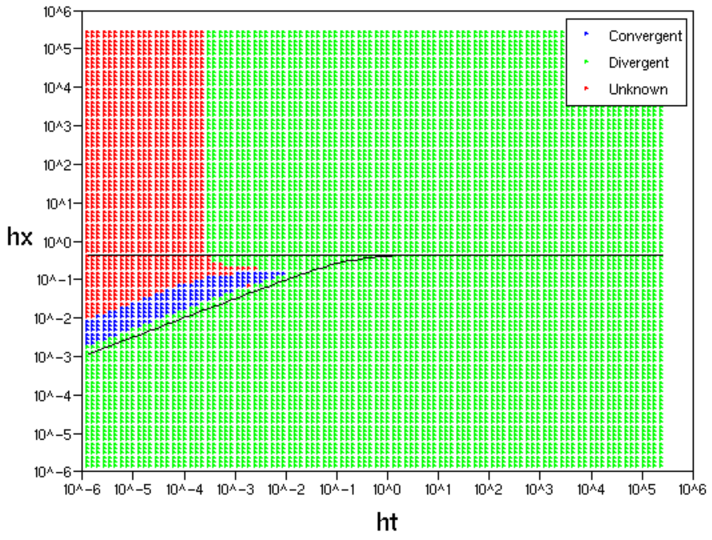

Fig. 5.4: Convergence map for the Forward Euler's function also containing boundary constraints in black.

5.2.2 Backward Euler's Convergence Boundaries

Now for the Backward Euler's equation I have the constraint:

$$1 = \left|1 + \frac{a_1 h_t}{1 - a_1 h_t} - \frac{2a_2 h_t}{(1 - a_1 h_t) h_x^2}\right| \tag{5.13}$$

Again I will break up the absolute value into two constraints with one constraint having a positive right side and one having a negative right side. First when the right side is positive side:

$$\begin{aligned} 1 &= 1 + \frac{a_1 h_t}{1 - a_1 h_t} - \frac{2a_2 h_t}{(1 - a_1 h_t) h_x^2} && (5.14) \\ \frac{a_1 h_t}{1 - a_1 h_t} &= \frac{2a_2 h_t}{(1 - a_1 h_t) h_x^2} && (5.15) \\ h_x^2 &= \frac{2a_2 h_t}{a_1 h_t} && (5.16) \\ h_x &= \sqrt{\frac{2a_2 h_t}{a_1 h_t}} && (5.17) \\ &= \sqrt{\frac{2a_2}{a_1}} && (5.18) \end{aligned}$$

Now, the second constraint with the right side set as negative becomes:

$$\begin{aligned} 1 &= -1 - \frac{a_1 h_t}{1 - a_1 h_t} + \frac{2a_2 h_t}{(1 - a_1 h_t) h_x^2} && (5.19) \\ 2 + \frac{a_1 h_t}{1 - a_1 h_t} &= \frac{2a_2 h_t}{(1 - a_1 h_t) h_x^2} && (5.20) \\ (1 - a_1 h_t) h_x^2 &= \frac{2a_2 h_t}{2 + \frac{a_1 h_t}{1 - a_1 h_t}} && (5.21) \\ &= \frac{2a_2 h_t}{\frac{2(1 - a_1 h_t)}{1 - a_1 h_t} + \frac{a_1 h_t}{1 - a_1 h_t}} && (5.22) \\ &= \frac{2a_2 h_t}{\frac{2(1 - a_1 h_t) + a_1 h_t}{1 - a_1 h_t}} && (5.23) \\ &= \frac{2a_2 h_t (1 - a_1 h_t)}{2(1 - a_1 h_t) + a_1 h_t} && (5.24) \\ h_x^2 &= \frac{2a_2 h_t}{2(1 - a_1 h_t) + a_1 h_t} && (5.25) \\ h_x &= \sqrt{\frac{2a_2 h_t}{2 - a_1 h_t}} && (5.26) \end{aligned}$$

This gives the following two boundary equations for Backward Euler's:

$$h_x = \sqrt{\frac{2a_2}{a_1}} \qquad (5.27)$$

$$h_x = \sqrt{\frac{2a_2h_t}{2 - a_1h_t}} \tag{5.28}$$

As you can see, the first constraint is a straight line and is the same for both Euler's equations. The difference is with the second constraint, which comes with an extra $(1 - a_1h_t)$ term for the Backward Euler's equation. When h_t is small, the entire term becomes close to one. When this is the case, it is the same second constraint as the Forward Euler's second constraint. The change happens when h_t becomes big. When it does, the term will cause the second constraint to shoot off towards infinity. Where the function goes to infinity is the boundary between the first and second quadrant, allowing the first quadrant to converge for the Backward Euler's function, where it does not for the Forward Euler's function.

Figure 5.5 is a convergence map of a Backward Euler's function with these two constraints.

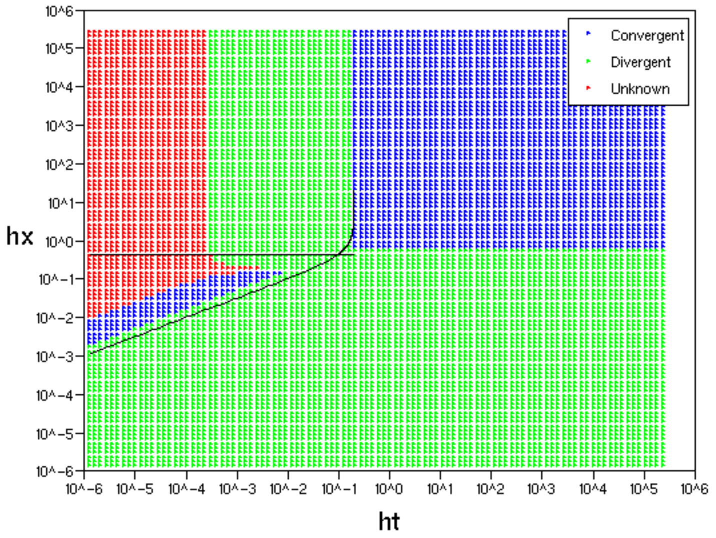

Fig. 5.5: Convergence map for the Backward Euler's function also containing boundary constraints in black.

5.2.3 a_3 Vertical Boundary

For both the Forward and Backward Euler's functions, a third boundary is made visible by varying the a_3 parameter. This parameter is not part of the poles constraint, and when it is changed it seems to create a near vertical line parallel to the h_x axis that is a boundary between the convergent and divergent areas. The boundary may not be a completely flat line because it appears to bevel in some instances, but only slightly. Because the poles constraint cannot account for the vertical divide, there must be some other inherent behavior creating what I will refer to in the rest of this thesis as the a_3 vertical boundary.

Table 5.1 contains the rightmost h_t value that converges for the specified a_3 value. This table demonstrates how changing a_3 can increase or decrease the amount of simulations that can converge with respect to h_t. The a_3 vertical boundary starts in the middle of the area depicted by the pole constraints when a_3 is zero. As a_3 increases, it slides closer to the right where the two pole constraints touch, (or in the case of Forward Euler's where they nearly touch) and once it reaches the tipping point, the a_3 vertical boundary begins to slide back in the other direction. It appears to be an even function because the h_t value for a_3 is equal to $-a_3$. Table 5.1 shows this behavior with a changing a_3 parameter. Figure 5.6, on page 45, illustrates the behavior with three example maps that are varied by a_3, at values one, ten, and fifty.

In Appendices A and B, on pages 67 and 88, I look at different possible reasons for the a_3 vertical boundary. Although I cannot find any explanations for the a_3 vertical boundary, I did find Observation A.3.1, on page 80, and Observation B.2.2, on page 94, which will be helpful in approximating the h_t value of the a_3 vertical boundary.

a_1	a_2	a_3	b_1	$\max(h_t)$
10	1	-100	1	0.0004922
10	1	-10	1	0.0326085
10	1	-5	1	0.0271738
10	1	1	1	0.0109205
10	1	5	1	0.0271738
10	1	10	1	0.0326085
10	1	50	1	0.0017637
10	1	100	1	0.0004922
10	1	500	1	0.0000185
10	1	1000	1	0.0000042

Tab. 5.1: Table depicts the largest h_t value that converges along the lower boundary of the convergence area within a convergence map for a changing a_3 parameter.

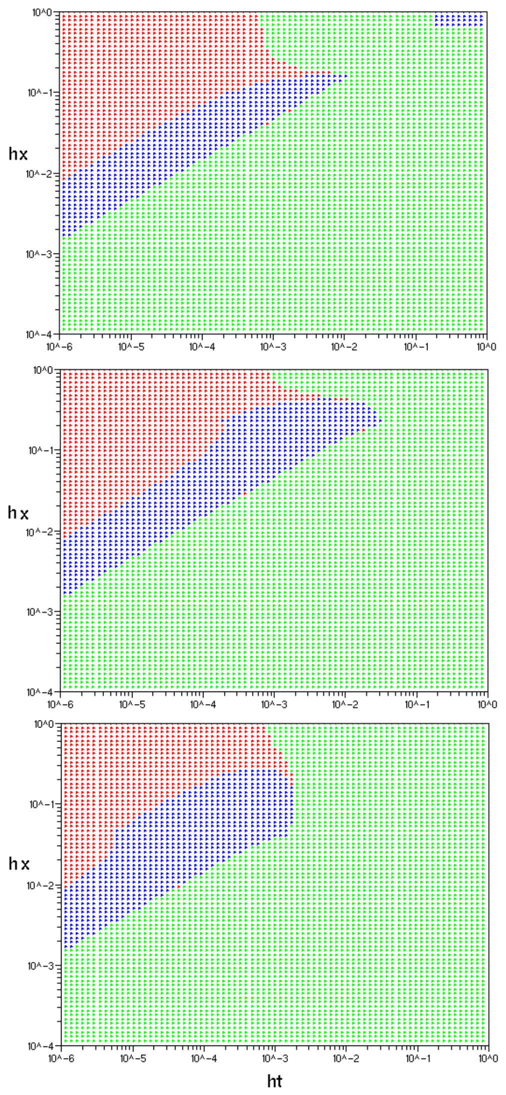

Fig. 5.6: Convergence maps for the Backward Euler's function demonstrating the a_3 vertical boundary.

In Section A.3.2, I state Observation A.3.1, which says that where the upper zeros boundary constraint crosses the lower poles constraint approximates where the a_3 vertical boundary is located. The approximation is not very good until the a_3 vertical boundary recedes back past its initial point, when $a_3 = 0$. Observation B.2.2 gives an approximate location for the initial h_t value of the a_3 vertical boundary at $\frac{0.1}{a_1}$. So the initial value $\frac{0.1}{a_1}$ can be used as an approximation until the intersection point of the zeros constraint and lower poles constraint is less than $\frac{0.1}{a_1}$. If it is less, then the intersection point can be used as an approximation. Either way, the approximation point will be multiplied by a safety buffer to ensure it is in the area of convergence. From the tests I have done, 60% is a good size for the safety buffer and will be used in my guidelines to pick h_t and h_x in Section 5.5 on page 50.

5.3 Boundary Errors

The following section will quantify the error for both the upper and lower theoretical boundaries compared against the real convergence map boundaries. The boundaries will only be measured for quadrant three, when both h_t and h_x are small, since that is where the interesting convergence area is at. The error was quantified for the Backward Euler's function but not the Forward Euler's function since for quadrant three they are nearly the same function and therefore have the same error.

5.3.1 Upper Boundary Error

Since the upper boundaries are straight horizontal lines, I can find the exact error for the upper boundaries. Table 5.2 provides that information for some of the simulations done. The error value is calculated by subtracting the value with the largest h_x that converges from the theoretical upper limit value $\sqrt{\frac{2a_2}{a_1}}$ and by then dividing that by the largest h_x that converges.

a_1	a_2	a_3	b_1	Err%
10	1	1	1	233.47566
100	1	1	1	201.18792
10	100	1	1	214.4662
10	1	100	1	156.51974
-10	1	100	1	UNDEF

Tab. 5.2: Table depicts the error value of the upper boundary.

5.3.2 Lower Boundary Error

Table 5.3 shows the average error for the lower theoretical boundary and the actual lower convergence boundary. The lower error is found by subtracting the actual convergence boundary by the theoretical boundary and dividing that by the theoretical boundary value. Notice that for variances of a_1 and a_2 the error appears to be a constant.

This table points to the fact that the error between the theoretical boundary and the actual boundary is approximately 45.6%, the average of the table values except where a_3 varies. The reason it varies for a_3 is that the end slightly dips down at the a_3 vertical boundary, but remains the same height for the rest of the lower boundary.

a_1	a_2	a_3	b_1	Avg Err%
0.1	1	0	1	46.42427
1	1	0	1	45.55027
10	1	0	1	46.38067
100	1	0	1	45.82837
1	0.01	0	1	45.24517
1	0.1	0	1	45.54375
1	10	0	1	45.23557
1	100	0	1	44.89877
1	1000	0	1	45.54375
1	1	0.1	1	45.70477
1	1	10	1	42.35140
1	1	40	1	39.14158

Tab. 5.3: Table depicts the error values between the lower theoretical boundary and actual boundary.

Calculating the error on this boundary is probably the most important, because, as seen in the next section, h_t and h_x combinations closest to the true lower boundary will converge faster. In Section 5.5, I will use this information to make recommendations on how to pick h_t and h_x for optimum convergence.

5.4 Rate of Convergence

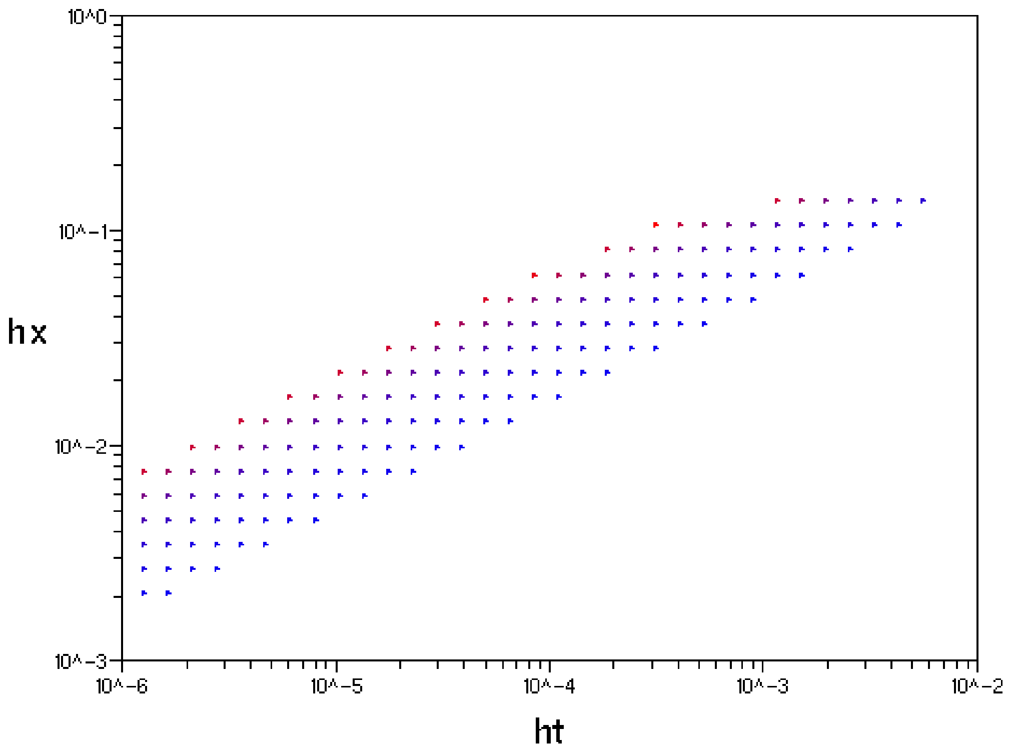

Fig. 5.7: Convergence map for the Backward Euler's function showing the speed of convergence through color variation.

Now that I have defined the area of convergence, another important aspect for practical concerns is how long it takes for a simulation to converge. Figure 5.7 shows those points in quadrant three of the convergence map that do converge. Their color has been altered slightly to show the points' colors range from red to blue. The pure red points took the longest to converge (about 3000 iterations) and the pure blue points converged quickly (about 200 iterations). From this information we can make a few recommendations.

The figure shows that as h_x becomes larger or as h_t becomes smaller it takes longer for those data points to converge. Therefore it is recommend that for the fastest speed, simulations should use h_t and h_x values closest to the lower boundary. They should not be on the boundary, however, because points directly on the boundary can do odd things and may take a long time to converge. This is due to the fact they are so close to the divergent section of the maps and the points are caught between converging or diverging.

5.5 Optimum Convergence Guidelines

In this section I will present guidelines on how to choose h_x and h_t so that the CA will converge and take the least number of iterations. As noted in Section 5.4, the closer the values h_x and h_t are to the lower boundary, the faster they will converge. So I will pick h_x using a modified form of the lower boundary equation, which is scaled by the error found in Section 5.3.2 to make sure it will be in the convergence area. To pick h_t, I will pick a value that is to the left of the a_3 vertical boundary that I was able to describe with the intersection of the lower poles constraint and the upper zeros constraint and its constant initial value in Section 5.2.3. From this data I have created the following guidelines in the form of an algorithm in Listing 5.1 to choose the best h_x and h_t values for convergence and speed. It should be noted that there is no theoretical proof given that the values suggested by the guidelines will converge, but they have been developed from observations of many simulations and are included to help those who wish to use the CA models I have created.

In the algorithm, lowerPoleHt(hx) is a predefined function for the lower pole constraint for either the Backward or Forward Euler's Equation that outputs the corresponding h_t parameter given h_x. For Forward Euler's it is defined as the following by solving Equation 5.12, on page 39, for h_t:

$$lowerPoleHt(hx) = \frac{2h_x^2}{2a_2 - a_1 h_x^2} \tag{5.29}$$

For Backward Euler's it is defined as the following by solving Equation 5.28, on page 42, for h_t:

$$lowerPoleHt(hx) = \frac{2h_x^2}{2a_2 + a_1 h_x^2} \tag{5.30}$$

The function lowerPoleHx(ht) is a function that given h_t will return h_x for the lower pole constraint. For Forward Euler's it is defined as the following from Equation 5.12, on page 39:

$$lowerPoleHx(ht) = \sqrt{\frac{2a_2 h_t}{2 + a_1 h_t}} \tag{5.31}$$

For Backward Euler's it is defined as the following from Equation 5.28, on page 42:

$$lowerPoleHx(ht) = \sqrt{\frac{2a_2 h_t}{2 - a_1 h_t}} \tag{5.32}$$

The function upperZeroHt(hx) is a predefined function for the upper zero boundary constraint for either the Backward or Forward Euler's Equation that outputs the corresponding h_t parameter given h_x.

For Forward Euler's it is defined as the following equation from Equation A.80, on page 82:

$$upperZeroHt(hx) = \frac{4h_x^2}{2a_2 - |a_3|h_x - 2a_1h_x^2} \tag{5.33}$$

For Backward Euler's it is defined as the following from Equation A.102, on page 85:

$$upperZeroHt(hx) = \frac{4h_x^2}{2a_2 - |a_3|h_x} \tag{5.34}$$

Also in the algorithm, the lower poles boundary is raised by 60%, which is greater than the 46.2% found in Section 5.3, to make sure the recommendation is within the area of convergence. Also by multiplying the maximum h_t value by 60% will ensure it is within the area of convergence, as mentioned in Section 5.2.3. I will search for the intersection from 10^{-15} times the initial max h_t value to 10^3 times the max h_t value, which should be well past where the constraints could intersect. Figures 5.8, 5.9 and 5.10 give examples of what parameters would be picked using the algorithm. These figures were plotted by using code in Listing C.8 on page 108.

Listing 5.1: Guidelines for picking h_t and h_x so that they will converge and the simulation speed will be relatively fast. Functions used in the algorithm are defined in Section 5.5. The algorithm code is in the Scilab language.

```
//returns ht,hx pair that will converge quickly given
//a1, a2, and a3 coefficents.
upSearch = (0.1/a1)*10^(3);
lowSearch = (0.1/a1)*10^(-15);
multiplier = 1.1;
safetyBuffer = .60;
safetyBuffer2 = 1.6;
```

```
htIntersect = -1;
htInit=0.1/a1;

//pick htMax
curHx = lowSearch;
while(curHx < upSearch) //search for instersection
  if(upperZeroHt(hx) - lowerPoleHt(hx) < 0) then
    htIntersect = lowerPoleHt(hx);
    break;
  end
  curHx = curHx * multiplier;
end

if((htIntersect == -1) || //they didn't intersect
  (htInit < htIntersect)) //htInit is lower
  htMax= htInit;
else
  htMax = htIntersect;
end

htMax = htMax * safetyBuffer;

//here user chooses ht such that ht <= htMax
```

```
ht = userInput();
//now use new ht to get corresponding hx
hx = safetyBuffer2  * lowerPoleHx(ht);
```

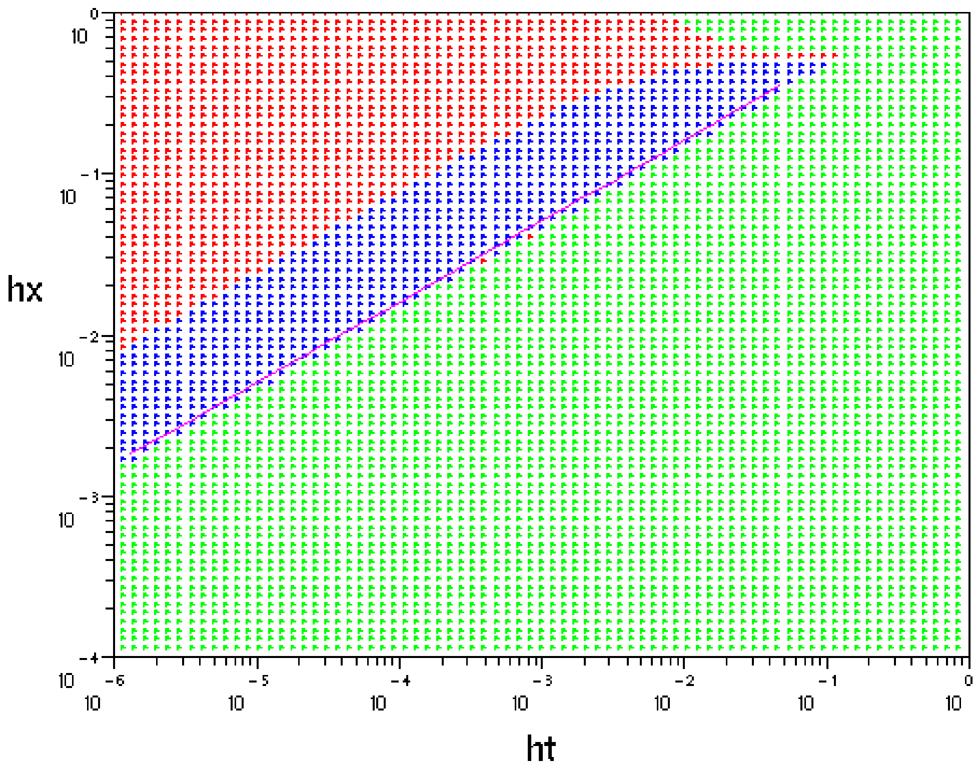

Fig. 5.8: Convergence map for the Backward Euler's function where the magenta line represents those values that the guidelines would pick for the following parameters: a_1=1, a_2=1, a_3=0, and b_1=1.

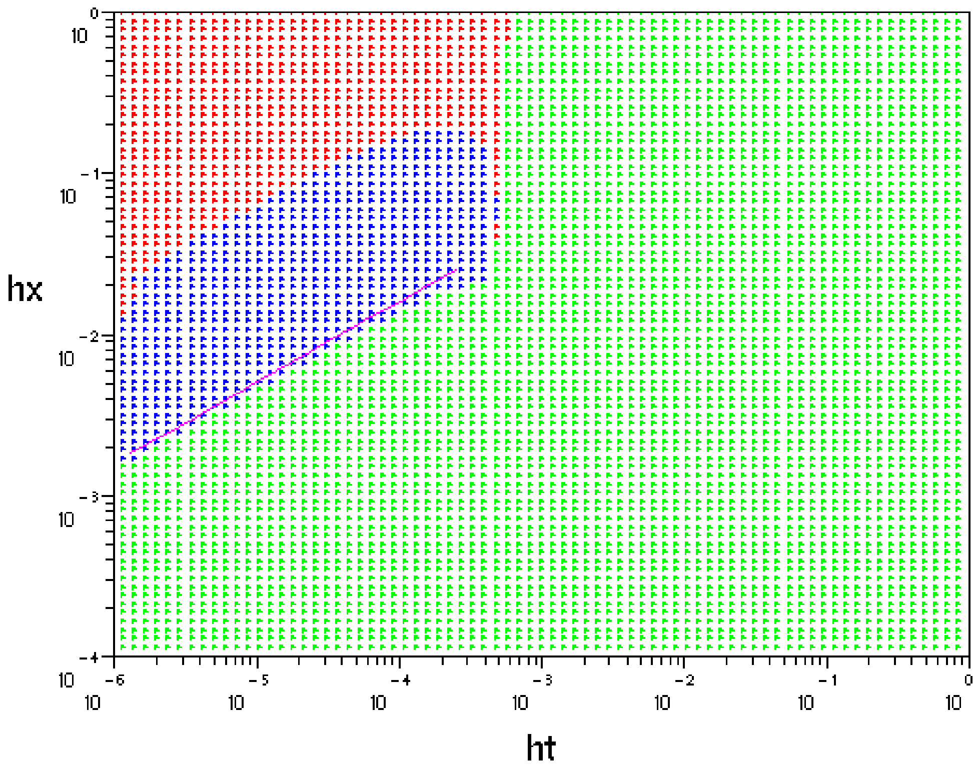

Fig. 5.9: Convergence map for the Backward Euler's function where the magenta line represents those values that the guidelines would pick for the following parameters: a_1=10, a_2=1, a_3=100, and b_1=1.

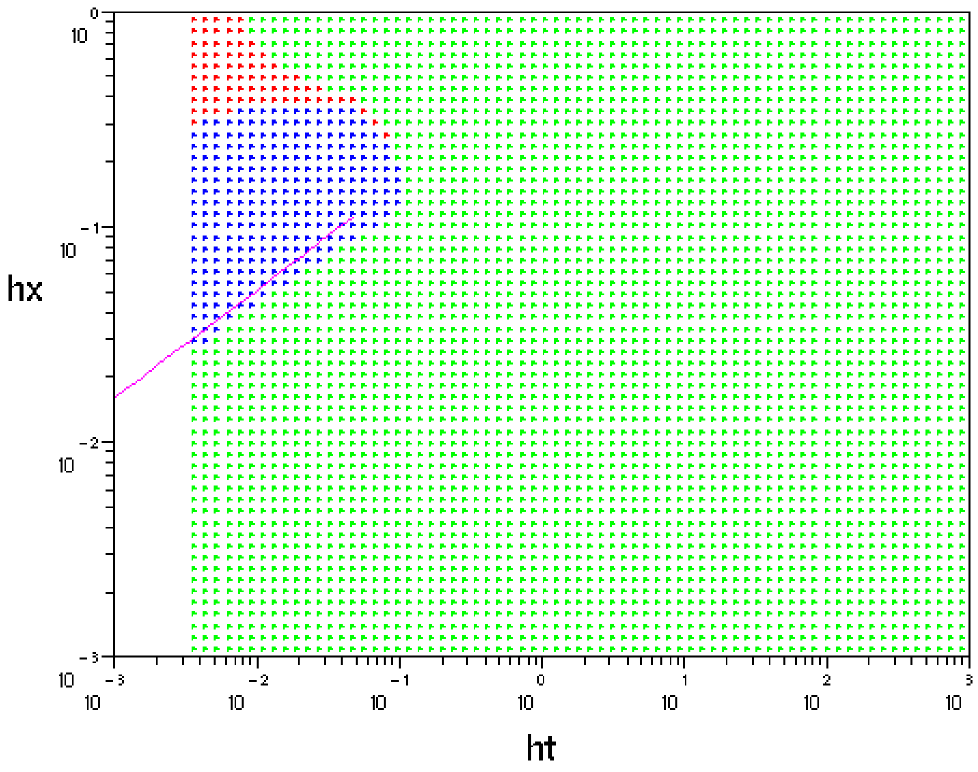

Fig. 5.10: Convergence map for the Backward Euler's function where the magenta line represents those values that the guidelines would pick for the following parameters: a_1=1, a_2=0.1, a_3=2.0, and b_1=1.

6. CONCLUSIONS

6.1 Summary of Findings

For this thesis I researched techniques used to simulate spatial partial differential equations with the cellular automata model, which may be beneficial to biologists. In Chapter 2 on page 5, I began with a literature review. I gave a few in depth examples of how biological equations can be modeled by partial differential equations (Section 2.2). In Section 2.3 I introduced some methods that could be used to discretize partial differential equations so they could be translated into a cellular automata model. Then I defined a cellular automata model in Section 2.4 and showed the advantages of using this model as opposed to using partial differential equations.

In Chapter 3 on page 17, I gave two methods, using the Forward and Backward Euler's methods, for converting a subset of partial differential equations into cellular automata. In Section 3.1, I explicitly defined the terms used to describe partial differential equations. In Section 3.2, I surveyed several biological partial differential equations defined in terms of both space and time. From these equations I found a general form that describes most of these biological equations, which is listed in Section 3.3 on page 21. The general form was defined in Equation 3.15 on page 21:

$$\frac{\partial u}{\partial t} = m(u_{i,j}) + \nabla_x^2 n(u_{i,j}) + \nabla_x o(u_{i,j}) \tag{6.1}$$

I then used discretization methods to solve the general differential equation form. Section 3.4.1 used the Forward Euler's method and Section 3.4.2 used the Backward Euler's method. The result of using discrete solvers were rules that can be used by CA to simulate the general form of the partial differential equation. The Forward Euler's method resulted in Equation 3.20 on page 22:

$$u_{i+1,j} = u_{i,j} + h_t \left(m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right) \tag{6.2}$$

The Backward Euler's method resulted in the following equation (Eq. 3.30, page 24):

$$u_{i+1,j} = u_{i,j} + \frac{h_t \left(m(u_{i,j}) + \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right)}{1 - h_t \frac{\partial m(u)}{\partial u}\Big|_{i,j}} \tag{6.3}$$

In Chapter 4 on page 25, I found where the Forward and Backward Euler's methods were stable for the general linear differential equation form. I did this by first defining the general linear form, taking the general form found in Section 3.3 and defining the coefficients so that they were only composed of linear terms of u. The resulting equation was (Eq. 4.1, page 26):

$$f(u) = a_1 u + b_1 + \nabla_x^2(a_2 u) + \nabla_x(a_3 u) \tag{6.4}$$

In Section 4.1.1, I then used the equation created from the Forward Euler's method on the general linear form and performed a Z-transform to find the poles of the equation. This resulted in the poles constraint equation (Eq. 4.17, page 28):

$$1 \;>\; \left| 1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2} \right| \tag{6.5}$$

and the zeros constraint equation(Eq. 4.15, page 28):

$$1 \;>\; \left| \frac{-1}{2b_1 h_x^2}((2a_2 - a_3 h_x)(U_{j-1}) + (2a_2 + a_3 h_x)(U_{j+1})) \right| \tag{6.6}$$

In Section 4.1.2, I used the Z-transform method on the Backward Euler's equation to find its constraints on stability. The zeros cosntraint was the same as the zeros constraint for Forward Euler's, but the poles constraint was the following(Eq. 4.30, page 31):

$$1 > \left|1 + \frac{a_1 h_t}{1 - a_1 h_t} - \frac{2a_2 h_t}{(1 - a_1 h_t)h_x^2}\right| \tag{6.7}$$

Finally in Section 4.2, I compared the two constraint equations for both Forward and Backward Euler's methods.

In Chapter 5, on page 33, I ran multiple simulations of the Forward and Backward General Linear formulas to map the areas of convergence with respect to space and time discretization sizes. In Section 5.1, I defined how I ran the CA simulations and gave some general descriptions of the graphs and the resulting convergence values. I found that the third quadrant, where h_x and h_t are small, contained an area of convergence that had interesting final convergence values. In Section 5.2, I graphed the boundaries found by the Z-transform constraints in Chapter 4, which formed the shape of the convergence area I was focused upon. I then found the average error between the theoretical constraints I graphed and the actual area of convergence in Section 5.3. In Section 5.4, I showed that the time to convergence was faster the closer the h_t,h_x pair was to the lower constraint.

In the appendices I attempted to explain a third convergence boundary I discovered. I called it the a_3 vertical boundary because it formed a vertical line in the h_t and h_x graphs and moved left or right depending upon the a_3 parameter. In Appendix A, I examined the zeros constraint and I tried to substitute in constant values for U_{j+1} and U_{j-1} in order to graph the zeros boundary. In Appendix B I examined whether stiffness could be the cause of the vertical a_3 boundary. I determined that stiffness is not a concern for a particular area of the convergence area.

Ultimately the two most important results of this thesis were the ability to convert a subset of partial differential equations to CA and the guidelines for convergence and simulation speed of the general linear form (Section 5.5, page 50). The guidelines came from two observations made in Appendices A and B. Observation A.3.1 on page 80 stated that the intersection between the lower poles constraint and the upper zeros constraint can be used to approximate the h_t value of the a_3 vertical boundary. Observation B.2.2 on page 94 stated that the maximum value of h_t will be $\frac{0.1}{a_1}$. The resulting guidelines are shown in Listing 5.1 on page 52. Code to plot the values that are picked by the guidelines is in Listing C.8 on page 108.

6.2 Future Work

The following subsections define new related problems that were brought to my attention as a result of this thesis.

6.2.1 Scaling u

In this thesis I kept the initial value of u constant at [1 2 3 4 5] in order to see how changing the parameters a_1, a_2, a_3, and b_1 would affect convergence. A problem that needs to be investigated is how increasing the number of values in u could affect the area of convergence.

I am using the value zero as boundary values in my CA simulations. Because u is so small the boundaries may be having more of an affect on the final outcome than it would when it is used in practice with possibly hundreds of values.

In one convergence map I created I did extend u to fifteen values with $a_1 = 1$, $a_2 = 1$, $a_3 = 0$. There were many more red points, which was to be expected because it should in general take more iterations for a fifteen value vector to converge that a five value vector. What was a little unexpected was that more points diverged as well, so the largest h_t was a little greater than 10^{-2} instead of 10^{-1} as predicted by Observation B.2.2 on page 94. This effect may indicate that the fifteen element vector is the same length as the five element vector. This means a smaller h_x is needed to converge for the fifteen element vector, and a correspondingly smaller h_t is needed to converge to keep the balance between the two. The correlation between number of initial u values and the maximum h_t and h_x values should be researched.

6.2.2 Proofs for Observations A.3.1 and B.2.2

In this thesis I made a couple of observations based upon repeated simulations. Observation A.3.1 on page 80 indicates that where the lower poles constraint and upper zeros boundary constraint cross may be the cause of vertical a_3 boundary. This may be a coincidence and that both the intersection and the boundary simply change at the same rate as a_3. Another possibility is that is that the intersection is causing some instability and causing points after it to diverge. The observation needs to be researched to see if this can be proven or disproven.

Observation B.2.2 on page 94 indicates that when $a_3 = 0$, the maximum h_t value will be about $\frac{0.1}{a_1}$. Although I saw this many times in different simulations, I was unable to give any theoretical reason why this was so. Also as discussed in Section 6.2.1, this Observation did not hold true with a larger data set. The Observation needs to be further research to understand the correlation between a_1 and the maximum h_t using multiple data sets to see if a more complex pattern emerges.

6.2.3 General Quadratic Form

In Chapters 4 and 5, I examined a general linear form of the partial differential equations subset. The most applicable form to the biological equations, however, is the general quadratic form, which contains a u^2 term. Explicitly the general quadratic form would like:

$$f(u) = d_1 u^2 + a_1 u + b_1 + \nabla_x^2(a_2 u) + \nabla_x(a_3 u) \tag{6.8}$$

Using the same techniques I used in this thesis with this equation (the Z-transform for stability analysis and creating convergence maps) would help with the eventual creation of software that can automatically determine the best h_x and h_t for many more biological equations than can be done currently with the general linear form.

6.2.4 Variable and Dependent Coefficients

In the general linear form I assume that a_1, a_2, and a_3 are constants that do not change from one time step to another. This however is not always the case. Not only may these parameters change over time, but their values may be dependent upon other simultaneous CA or differential equations.

The desert vegetation pattern equations are a perfect example of partial differential equations that are interdependent:

$$\frac{\partial n}{\partial t} = \frac{yw}{1+\sigma w}n - n^2 - \mu n + \nabla^2 n \tag{6.9}$$

$$\frac{\partial w}{\partial t} = p - (1-\rho n)w - w^2 n + \delta\nabla^2(w - \beta n) - v\frac{\partial(w-\alpha n)}{\partial x} \tag{6.10}$$

The equation for the biomass variable n contains the variable water w which changes with time. Similarly the equation for w depends upon the value of n. This means the area of convergence will also be dynamic and finding how the area will change will be critical to know how to simulate these equations properly.

6.2.5 Long Run Stability and a_3

During my research I was unable to explain and predict with theory the a_3 vertical boundary. I tried several different avenues in Appendices A and B, but in the end I used observations to describe the constraint. Understanding why a_3 has such an effect on convergence is important and should be investigated further. One possible reason is that the long term stability of the simulation is affected greatly by this parameter. Most computer simulations can last for a large number of iterations, but at some point tend to go inexplicable haywire and unstable even after stability appears to have been achieved. It may be possible that the a_3 parameter is greatly affecting the length of time that the simulation can go before it naturally goes unstable, cutting the simulation time too short before it can converge on a value. The current area of convergence should be tested to determine in general what this length of time is and how a_3 may be affecting it.

6.2.6 Parallel Implementation

As mentioned earlier, an advantage of the cellular automata model is that it lends itself to data parallelism. The grid of cells in the CA can be split up between processors and network nodes. The ability to parallelize a scientific simulation in order for it to take advantage of grid computing is critical for large scale simulations. An efficient means of parallelizing the models I have proposed should be explored.

6.2.7 Convergence Values and Simulation Errors

Another problem that should be looked into is the correlation between the general linear parameters and the final values of convergence. As noted in Section 5.1.2, on page 36, the area of convergence in the third quadrant has converges to values of interest. In this area simulations that use the same h_x parameter will have nearly the same convergence values, which means how time is discretized does not matter but how space is discretized does. The correlation between size discretization and the convergence values should be studied. Also, I have noticed that b_1 does not seem to affect the area of convergence, but it has an impact on the convergence values.

In my thesis I only studied the CA simulations in one space dimension. Biological Equations need to use two and possible three dimensions in order to be properly simulated. This will add extra discretization parameters like h_y and h_z that need to be studied to see how they will affect the area of convergence.

Lastly, now that I have established areas of converge for linear terms, the accuracy of the CA model I have created needs to be compared against the actual partial differential equation model. The error between the two needs to be quantified so that a biologist using the CA models know for how many iterations the CA will be reliable. Once this is done a program can be made to help simulate CA models for biologists.

APPENDIX A

EXAMINING THE ZEROS CONSTRAINT

In subsection 4.1.1, it was shown that for the general Linear Forward Euler's Equation (Eq. 4.5),

$$\begin{aligned} u_{i+1,j} &= u_{i,j} + h_t \Bigg(a_1 u_{i,j} + b_1 + \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \\ &\quad \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x} \Bigg) \end{aligned} \tag{A.1}$$

that there will be one poles constraint (Eq. 4.17) and one zeros constraint (Eq. 4.15). The zeros constraint is shown below:

$$1 > \left| \frac{-1}{2b_1 h_x^2} ((2a_2 - a_3 h_x)(U_{j-1}) + (2a_2 + a_3 h_x)(U_{j+1})) \right| \tag{A.2}$$

The problem is that this constraint cannot be graphed or solved in terms of h_x or h_t in order to see how the stability constraints affect the convergence area because of the two non-constant values U_{j-1} and U_{j+1}. Therefore I need to substitute these values with constant approximations, which is shown in the first section of this appendix.

This leads to two further constraints developed within this appendix. In Section A.2, on page 70, I will find a second pole constraint that is extremely similar to the original pole constraint found in Equation 4.17. In Section A.3, on page 73, I will develop the zero constraint in terms of the boundary values of u_i to attempt to explain the vertical a_3 boundary found in Chapter 5. It should be noted that the efforts to replace U_{j-1} and U_{j+1} ultimately fail to explain the vertical a_3 constraint. However in Section A.3.2 on page 78, an observation is made that can be used to describe the boundary.

A.1 Eliminating U_{j-1} and U_{j+1} for Forward Euler's

Now I need to substitute the values U_{j-1} and U_{j+1} to try to find a constraint with only constants. I can use the previous equation for U_j (Eq. 4.12, page 27) for this substitution. This would, however, result in the values U_{j-2} and U_{j+2} being part of the equation. Since I am just trying to estimate for these values, I will set both of these values to zero so that the equation will only be in terms of U_j and other constant values. Therefore the substitutions for these equations are the following:

$$U_{j-1} = \frac{\frac{h_t}{2h_x^2}((2a_2 + a_3h_x)(z^{-1}U_j) + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{A.3}$$

and

$$U_{j+1} = \frac{\frac{h_t}{2h_x^2}((2a_2 - a_3h_x)(z^{-1}U_j) + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{A.4}$$

Now I will substitute these into the two terms that contain U_{j-1} and U_{j+1} in the previous equation. I am only going to change those two terms at the moment, in order to cancel terms and clean them up before putting them into the final equation. Also I will define $c_1 = \frac{z^{-2}h_t}{(2h_x^2)\left(1-\frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t}+2a_1h_x^2-4a_2\right)(z^{-1})\right)}$, for readability:

$$(2a_2 - a_3h_x)(z^{-1}U_{j-1})+ \tag{A.5}$$

$$\begin{aligned}(2a_2 + a_3h_x)(z^{-1}U_{j+1}) &= (2a_2 - a_3h_x)\left(c_1((2a_2 + a_3h_x)U_j+\right.\\ &\quad \left.2b_1h_x^2)\right) + \\ &\quad (2a_2 + a_3h_x)\left(c_1((2a_2 - a_3h_x)U_j+\right.\\ &\quad \left.2b_1h_x^2)\right) \qquad (A.6)\\ &= 8a_2^2c_1U_j - 2a_3^2h_x^2c_1U_j + 8a_2b_1h_x^2c_1 \qquad (A.7)\\ &= c_1(8a_2^2 - 2a_3^2h_x^2)U_j + 8a_2b_1h_x^2c_1 \qquad (A.8)\end{aligned}$$

I substitute in the result of Equation A.8 back into Equation 4.12 for the terms $(2a_2 - a_3h_x)(z^{-1}U_{j-1}) + (2a_2 + a_3h_x)(z^{-1}U_{j+1})$. I then solve for U_j again:

$$U_j = \frac{\frac{h_t}{2h_x^2}(c_1(8a_2^2 - 2a_3^2h_x^2)U_j + 8a_2b_1h_x^2c_1 + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{A.9}$$

I again gather U_j terms on the left side:

$$\left(1 - \frac{\frac{h_t}{2h_x^2}(c_1(8a_2^2 - 2a_3^2h_x^2))}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)}\right)U_j = \frac{\frac{h_t}{2h_x^2}(8a_2b_1h_x^2c_1 + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{A.10}$$

Now divide by the coefficient of U_j and clean things up:

$$U_j = \frac{\frac{h_t}{2h_x^2}(8a_2b_1h_x^2c_1 + 2b_1h_x^2)}{1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1}) - \frac{h_t}{2h_x^2}(c_1)(8a_2^2 - 2a_3^2h_x^2)} \tag{A.11}$$

Now I substitute c_1 back in and manipulate the terms so that the exponents of z are all positive to find the poles and the zeros of the equation. Also, because the same set of terms are repeated, I am going to let $c_2 = \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)$. This gives:

$$\begin{aligned}
U_j &= \frac{\frac{h_t}{2h_x^2}((8a_2b_1h_x^2)\left(\frac{z^{-2}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right) + 2b_1h_x^2)}{1 - c_2(z^{-1}) - \frac{h_t}{2h_x^2}\left(\frac{z^{-2}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)(8a_2^2 - 2a_3^2h_x^2)} && \text{(A.12)}\\
&= \frac{\frac{h_t}{2h_x^2}((8a_2b_1h_x^2)\left(\frac{z^{-1}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right) + 2b_1h_x^2z)}{z - c_2 - \frac{h_t}{2h_x^2}\left(\frac{z^{-1}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)(8a_2^2 - 2a_3^2h_x^2)} && \text{(A.13)}\\
&= \frac{\frac{h_t}{2h_x^2}((8a_2b_1h_x^2)\left(\frac{h_t}{(2h_x^2)(z-c_2)}\right) + 2b_1h_x^2z)}{z - c_2 - \frac{h_t}{2h_x^2}\left(\frac{h_t}{(2h_x^2)(z-c_2)}\right)(8a_2^2 - 2a_3^2h_x^2)} && \text{(A.14)}\\
&= \frac{\frac{h_t}{2h_x^2}(\left(\frac{8a_2b_1h_th_x^2}{(2h_x^2)(z-c_2)}\right) + 2b_1h_x^2z)}{z - c_2 - \left(\frac{h_t^2(8a_2 - 2a_3^2h_x^2)}{(4h_x^4)(z-c_2)}\right)} && \text{(A.15)}\\
&= \frac{\frac{h_t}{4h_x^4}\left(8a_2b_1h_th_x^2 + 4b_1h_x^4\left(z^2 - c_2z\right)\right)}{\left(z - c_2 - \left(\frac{h_t^2(8a_2^2 - 2a_3^2h_x^2)}{(4h_x^4)(z-c_2)}\right)\right)(z - c_2)} && \text{(A.16)}
\end{aligned}$$

From this equation I can come up with three constraints, one from the numerator of the equation and two constraints from the two factors in the denominator. The factor $(z - c_2)$ will actually lead to the original poles constraint I found previously (Eq. 4.17). Therefore I will focus on the other denominator factor to obtain the pole value and our second pole constraint.

A.2 Second Pole Constraint

I will obtain this second pole constraint by setting the factor $z - c_2 - \left(\frac{h_t^2(8a_2^2 - 2a_3^2h_x^2)}{(4h_x^4)(z-c_2)}\right)$, from the denominator of Equation A.16, equal to zero and solving for z. The constraint I create will be that z value set to less than one:

$$
\begin{aligned}
0 &= z - c_2 - \left(\frac{h_t^2(8a_2^2 - 2a_3^2h_x^2)}{(4h_x^4)\,(z - c_2)}\right) && \text{(A.17)}\\
&= z(4h_x^4)\,(z - c_2) - c_2(4h_x^4)\,(z - c_2) - h_t^2(8a_2^2 - 2a_3^2h_x^2) && \text{(A.18)}\\
&= 4h_x^4z^2 - 8h_x^4c_2z + 4h_x^4c_2^2 - h_t^2(8a_2^2 - 2a_3^2h_x^2) && \text{(A.19)}
\end{aligned}
$$

Using the quadratic equation I then get the following for z:

$$
\begin{aligned}
z &= \frac{8h_x^4c_2 \pm \sqrt{64h_x^8c_2^2 - 16h_x^4(4h_x^4c_2^2 - h_t^2(8a_2^2 - 2a_3^2h_x^2))}}{8h_x^4} && \text{(A.20)}\\
&= c_2 \pm \sqrt{64h_x^8c_2^2 - 16h_x^4(4h_x^4c_2^2 - h_t^2(8a_2^2 - 2a_3^2h_x^2))} && \text{(A.21)}\\
&= c_2 \pm \sqrt{128h_x^4h_t^2a_2^2 - 32h_x^6h_t^2a_3^2} && \text{(A.22)}\\
&= c_2 \pm 4h_th_x^2\sqrt{8a_2^2 - 2h_x^2a_3^2} && \text{(A.23)}\\
&= \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right) \pm 4h_th_x^2\sqrt{8a_2^2 - 2h_x^2a_3^2} && \text{(A.24)}\\
&= 1 + a_1h_t - \frac{2a_2h_t}{h_x^2} \pm 4h_th_x^2\sqrt{8a_2^2 - 2h_x^2a_3^2} && \text{(A.25)}\\
& && \text{(A.26)}
\end{aligned}
$$

This gives the final poles constraint that we can now graph against the original poles constraint:

$$1 > \left|1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2} \pm 4h_t h_x^2 \sqrt{8a_2^2 - 2h_x^2 a_3^2}\right| \tag{A.27}$$

A.2.1 Graphing Second Pole Constraint

The second poles constraint for Forward Euler's is Equation A.27:

$$1 > \left|1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2} \pm 4h_t h_x^2 \sqrt{8a_2^2 - 2h_x^2 a_3^2}\right| \tag{A.28}$$

I again produce two constraints from the absolute value, one from when the right side is positive and one from when its negative. One might be tempted to say I have four constraints because of the $\pm$, but when I square the right most factor to solve for h_x, it will not matter whether the factor is positive or negative. So now I will first solve for the constraint when the right side is positive and set it equal to one:

$$\begin{aligned}
1 &= 1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2} \pm \\
&\quad 4h_t h_x^2 \sqrt{8a_2^2 - 2h_x^2 a_3^2} && \text{(A.29)}\\
-a_1 h_t + \frac{2a_2 h_t}{h_x^2} &= \pm 4h_t h_x^2 \sqrt{8a_2^2 - 2h_x^2 a_3^2} && \text{(A.30)}\\
a_1^2 h_t^2 - \frac{4a_2 a_1 h_t^2}{h_x^2} + \frac{4a_2^2 h_t^2}{h_x^4} &= 16h_t^2 h_x^4 (8a_2^2 - 2h_x^2 a_3^2) && \text{(A.31)}\\
a_1^2 - \frac{4a_2 a_1}{h_x^2} + \frac{4a_2^2}{h_x^4} &= 16h_x^4 (8a_2^2 - 2h_x^2 a_3^2) && \text{(A.32)}\\
a_1^2 - \frac{4a_2 a_1}{h_x^2} + \frac{4a_2^2}{h_x^4} &= 128h_x^4 a_2^2 - 32h_x^6 a_3^2 && \text{(A.33)}\\
32a_3^2 h_x^{10} - 128a_2^2 h_x^8 + a_1^2 h_x^4 - 4a_2 a_1 h_x^2 + 4a_2^2 &= 0 && \text{(A.34)}
\end{aligned}$$

From Equation A.34 I see that h_t has completely canceled out leaving only h_x. This means that if I were to graph the equation with our convergence maps from Chapter

5, the result would be a horizontal line. However I am looking for a constraint that is vertical to explain the a_3 vertical constraint. Therefore I will stop developing this equation since it is not useful.

Next I will set the right side of Equation A.28 to be negative and make it equal to 1. Then I will solve for h_t to get another constraint:

$$\begin{aligned} 1 &= -1 - a_1 h_t + \frac{2a_2 h_t}{h_x^2} \mp \\ &\quad 4h_t h_x^2 \sqrt{8a_2^2 - 2h_x^2 a_3^2} \qquad (A.35) \\ 2 + a_1 h_t - \frac{2a_2 h_t}{h_x^2} &= \mp 4h_t h_x^2 \sqrt{8a_2^2 - 2h_x^2 a_3^2} \qquad (A.36) \\ 4 + 4a_1 h_t - \frac{8a_2 h_t}{h_x^2} + a_1^2 h_t^2 - \frac{4a_1 a_2 h_t^2}{h_x^2} + \frac{4a_2^2 h_t^2}{h_x^4} &= 128h_x^4 a_2^2 - 32h_x^6 a_3^2 \qquad (A.37) \\ 32a_3^2 h_x^{10} - 128a_2^2 h_x^8 + (4 + 4a_1 h_t + a_1^2 h_t^2)h_x^4 - & \\ (8a_2 h_t + 4a_1 a_2 h_t^2)h_x^2 + 4a_2^2 h_t^2 &= 0 \qquad (A.38) \\ (a_1^2 h_x^4 - 4a_1 a_2 h_x^2 + 4a_2^2)h_t^2 + & \\ (4a_1 h_x^4 - 8a_2 h_x^2)h_t + & \\ 32a_3^2 h_x^{10} - 128a_2^2 h_x^8 + 4h_x^4 &= 0 \qquad (A.39) \end{aligned}$$

Now I solve using the quadratic equation for h_t and get the following:

$$h_t = \frac{-d_2 \pm \sqrt{d_2^2 - 4d_1 d_3}}{2d_1} \qquad (A.40)$$

where

$$\begin{aligned} d_1 &= a_1^2 h_x^4 - 4a_1 a_2 h_x^2 + 4a_2^2 \qquad (A.41) \\ d_2 &= 4a_1 h_x^4 - 8a_2 h_x^2 \qquad (A.42) \\ d_3 &= 32a_3^2 h_x^{10} - 128a_2^2 h_x^8 + 4h_x^4 \qquad (A.43) \end{aligned}$$

When graphed this equation provides little extra information about the shape of the convergence maps because the second pole constraint is so similar to the original. It does vary slightly when a_3 is close to zero, but not enough to explain the a_3 vertical constraint seen in the convergence maps as a_3 becomes large. Figures A.1 and A.2 show the equation graphed with a_3=100 and a_3=1.

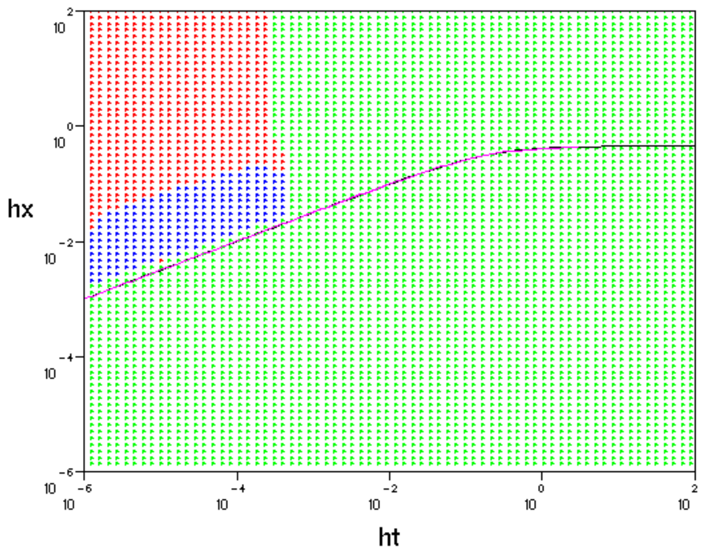

Fig. A.1: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new pole constraints in magenta at a_3=100.

A.3 Boundary Zero Constraint

Another area I investigated concerning the substituting of U_{j-1} and U_{j+1} was the newly created zeros boundary. Unfortunately, the numerator of Equation A.16 does not include the term a_3 at all, meaning the zeros constraint would also not contain a_3. Since the vertical a_3 constraint is what I am trying to explain, the zeros constraint would not help at all.

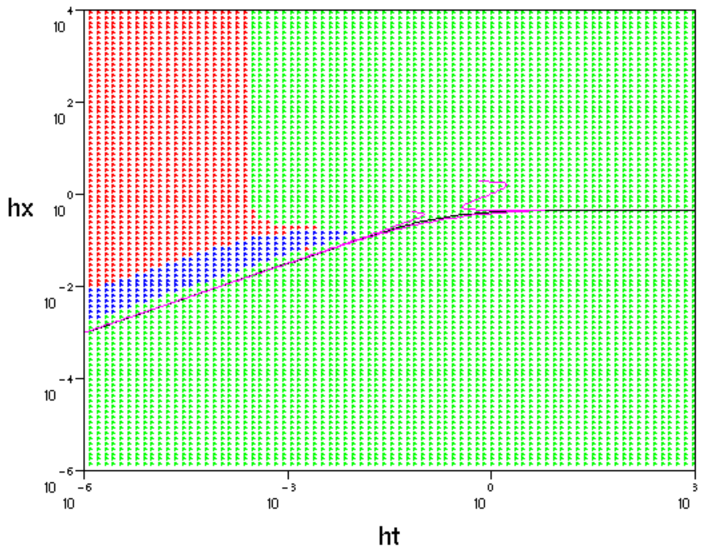

Fig. A.2: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new pole constraints in magenta at a_3=1.

However, since I use zero as the boundary value for u in our simulations, for the left most cell of u, $U_{j-1} = 0$, and for the rightmost cell of u, $U_{j+1} = 0$. If I take this into account then an extra term is not canceled out when substituting in for $(2a_2 - a_3h_x)(z^{-1}U_{j-1})(2a_2 + a_3h_x)(z^{-1}U_{j+1})$. For the left cell, I get an extra $2a_3b_1h_x^3$ term and for the right most cell an extra $-2a_3b_1h_x^3$ term. Therefore if I add in an extra $\pm 2a_3b_1h_x^3c_1$ term and halve the coefficients of the rest of the substitution terms (since $U_{j-1} = 0$ or $U_{j+1} = 0$, the other terms coefficients no longer double from addition) I get the following:

$$U_j = \frac{\frac{h_t}{2h_x^2}(c_1(4a_2^2 - a_3^2h_x^2)U_j + 4a_2b_1h_x^2c_1 \pm 2a_3b_1h_x^3c_1 + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{A.44}$$

I again gather U_j terms on the left side of the equations to solve for U_j like before:

$$(1- \tag{A.45}$$

$$\frac{\frac{h_t}{2h_x^2}(c_1(4a_2^2 - a_3^2h_x^2))}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)}\Bigg)U_j = \frac{\frac{h_t}{2h_x^2}(4a_2b_1h_x^2c_1 \pm 2a_3b_1h_x^3c_1 + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)}$$

Now divide by the coefficient of U_j and clean things up:

$$U_j = \frac{\frac{h_t}{2h_x^2}(4a_2b_1h_x^2c_1 \pm 2a_3b_1h_x^3c_1 + 2b_1h_x^2)}{1 - \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1}) - \frac{h_t}{2h_x^2}(c_1)(4a_2^2 - a_3^2h_x^2)} \tag{A.46}$$

Now I substitute c_1 back in and manipulate the terms so that the exponents of z are all positive to find the poles and the zeros of the equation. Also, because the same set of terms are repeated, I am going to let $c_2 = \frac{h_t}{2h_x^2}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)$. This gives:

$$U_j = \frac{\frac{h_t}{2h_x^2}\left((4a_2b_1h_x^2)\left(\frac{z^{-2}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right) \pm 2a_3b_1h_x^3\left(\frac{z^{-2}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)\right)}{1-c_2(z^{-1})-\frac{h_t}{2h_x^2}\left(\frac{z^{-2}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)(4a_2^2-a_3^2h_x^2)} + \frac{\frac{h_t}{2h_x^2}(2b_1h_x^2)}{1-c_2(z^{-1})-\frac{h_t}{2h_x^2}\left(\frac{z^{-2}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)(4a_2^2-a_3^2h_x^2)} \quad (A.47)$$

$$= \frac{\frac{h_t}{2h_x^2}\left((4a_2b_1h_x^2)\left(\frac{z^{-1}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right) \pm 2a_3b_1h_x^3\left(\frac{z^{-1}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)\right)}{z-c_2-\frac{h_t}{2h_x^2}\left(\frac{z^{-1}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)(4a_2^2-a_3^2h_x^2)} + \frac{\frac{h_t}{2h_x^2}(2b_1h_x^2z)}{z-c_2-\frac{h_t}{2h_x^2}\left(\frac{z^{-1}h_t}{(2h_x^2)(1-c_2(z^{-1}))}\right)(4a_2^2-a_3^2h_x^2)} \quad (A.48)$$

$$= \frac{\frac{h_t}{2h_x^2}((4a_2b_1h_x^2)\left(\frac{h_t}{(2h_x^2)(z-c_2)}\right) \pm 2a_3b_1h_x^3\left(\frac{h_t}{(2h_x^2)(z-c_2)}\right) + 2b_1h_x^2z)}{z-c_2-\frac{h_t}{2h_x^2}\left(\frac{h_t}{(2h_x^2)(z-c_2)}\right)(4a_2^2-a_3^2h_x^2)} \quad (A.49)$$

$$= \frac{\frac{h_t}{2h_x^2}(\left(\frac{4a_2b_1h_th_x^2}{(2h_x^2)(z-c_2)}\right) \pm \left(\frac{2a_3b_1h_x^3h_t}{(2h_x^2)(z-c_2)}\right) + 2b_1h_x^2z)}{z-c_2-\left(\frac{h_t^2(4a_2-a_3^2h_x^2)}{(4h_x^4)(z-c_2)}\right)} \quad (A.50)$$

$$= \frac{\frac{h_t}{4h_x^4}\left(4a_2b_1h_th_x^2 \pm 2a_3b_1h_x^3h_t + 4b_1h_x^4\left(z^2-c_2z\right)\right)}{\left(z-c_2-\left(\frac{h_t^2(4a_2^2-a_3^2h_x^2)}{(4h_x^4)(z-c_2)}\right)\right)(z-c_2)} \quad (A.51)$$

$$= \frac{\frac{b_1h_t}{2h_x^2}\left(2a_2h_t \pm a_3h_xh_t + 2h_x^2\left(z^2-c_2z\right)\right)}{\left(z-c_2-\left(\frac{h_t^2(4a_2^2-a_3^2h_x^2)}{(4h_x^4)(z-c_2)}\right)\right)(z-c_2)} \quad (A.52)$$

To find the zeros constraint I will set the numerator equal to zero and solve for z:

$$0 = \frac{b_1h_t}{2h_x^2}\left(2a_2h_t \pm a_3h_xh_t + 2h_x^2\left(z^2-c_2z\right)\right) \quad (A.53)$$

$$= (2a_2h_t \pm a_3h_xh_t + 2h_x^2(z^2-c_2z)) \quad (A.54)$$

Using the quadratic equation I get the following constraint by setting the z value to less than 1:

$$1 > \left| \frac{-d_2 \pm \sqrt{d_2^2 - 4d_1d_3}}{2d_1} \right| \tag{A.55}$$

where

$$\begin{aligned} d_1 &= 2h_x^2 && \text{(A.56)} \\ d_2 &= -2h_x^2c_2 && \text{(A.57)} \\ d_3 &= 2a_2h_t \pm a_3h_xh_t && \text{(A.58)} \end{aligned}$$

A.3.1 Graphing the Boundary Zero Constraint When the Right Side is Positive

First I will graph the zeros constraint when the right side is positive and rearrange the equation so that things cancel out before I substitute the *d* variables back in:

$$\begin{aligned} 1 &= \frac{-d_2 \pm \sqrt{d_2^2 - 4d_1d_3}}{2d_1} && \text{(A.59)} \\ 2d_1 + d_2 &= \pm\sqrt{d_2^2 - 4d_1d_3} && \text{(A.60)} \\ 4d_1^2 + 4d_1d_2 + d_2^2 &= d_2^2 - 4d_1d_3 && \text{(A.61)} \\ 4d_1^2 + 4d_1d_2 &= -4d_1d_3 && \text{(A.62)} \\ d_1 + d_2 &= -d_3 && \text{(A.63)} \end{aligned}$$

I will substitute back in the d variables and the c_2 variable. Then I solve for h_x to graph the equation:

$$\begin{aligned} 2h_x^2 - 2h_x^2 c_2 &= -2a_2 h_t \mp a_3 h_x h_t && \text{(A.64)} \\ 2h_x^2 - h_t(\frac{2h_x^2}{h_t} + 2a_1 h_x^2 - 4a_2) &= -2a_2 h_t \mp a_3 h_x h_t && \text{(A.65)} \\ 2h_x^2 - 2h_x^2 - 2a_1 h_x^2 h_t + 4a_2 h_t &= -2a_2 h_t \mp a_3 h_x h_t && \text{(A.66)} \\ -2a_1 h_x^2 + 4a_2 &= -2a_2 \mp a_3 h_x && \text{(A.67)} \\ & && \text{(A.68)} \end{aligned}$$

To solve for h_x, I can use the quadratic equation again:

$$h_x = \frac{-a_3 \pm \sqrt{a_3^2 + 48a_1 a_2}}{-4a_1} \tag{A.69}$$

When I graph these horizontal lines, a couple do not show within the graphs because they are negative. Unfortunately, these horizontal lines do not show why there is a vertical a_3 constraint. Figures A.3 and A.4 show the equation graphed with a_3=100 and a_3=1. The lines spread out as a_3 grows large but seem to have no bearing on the a_3 vertical constraint.

A.3.2 Graphing the Boundary Zero Constraint When the Right Side is Negative

Next I will graph the zeros constraint (Eq. A.55) when the right side is negative and rearrange the equation so that things cancel out before I substitute the d variables back in:

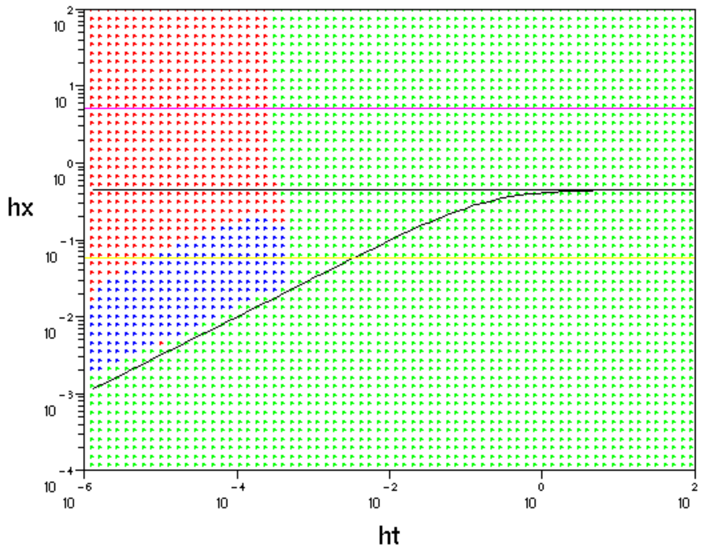

Fig. A.3: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new zeros constraints in magenta and yellow at a_3=100.

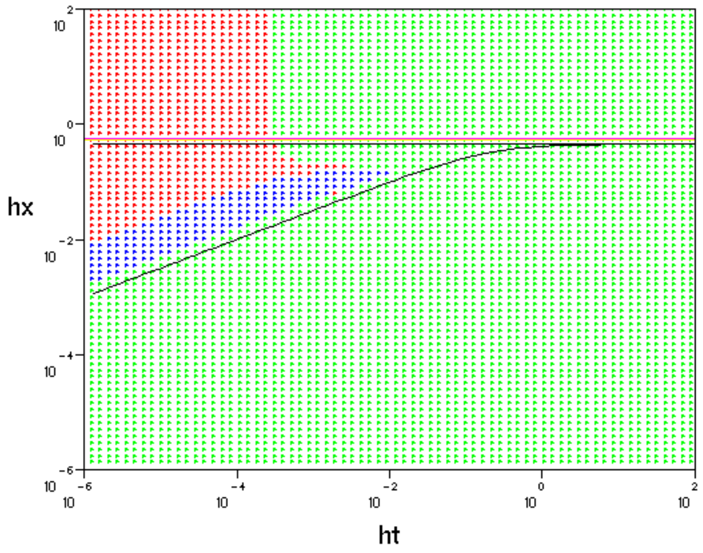

Fig. A.4: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new zeros constraints in magenta and yellow at a_3=1.

$$\begin{aligned}
1 &= \frac{d_2 \mp \sqrt{d_2^2 - 4d_1d_3}}{2d_1} && \text{(A.70)}\\
2d_1 - d_2 &= \mp\sqrt{d_2^2 - 4d_1d_3} && \text{(A.71)}\\
4d_1^2 - 4d_1d_2 + d_2^2 &= d_2^2 - 4d_1d_3 && \text{(A.72)}\\
4d_1^2 - 4d_1d_2 &= -4d_1d_3 && \text{(A.73)}\\
d_1 - d_2 &= -d_3 && \text{(A.74)}
\end{aligned}$$

Now I will substitute back in the d variables and then the c_2 variable. Then I solve for h_t to graph the equation:

$$\begin{aligned}
2h_x^2 + 2h_x^2c_2 &= -2a_2h_t \mp a_3h_xh_t && \text{(A.75)}\\
2h_x^2 + h_t(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2) &= -2a_2h_t \mp a_3h_xh_t && \text{(A.76)}\\
2h_x^2 + 2h_x^2 + 2a_1h_x^2h_t - 4a_2h_t &= -2a_2h_t \mp a_3h_xh_t && \text{(A.77)}\\
4h_x^2 &= 2a_2h_t \mp a_3h_xh_t - 2a_1h_x^2h_t && \text{(A.78)}\\
h_t &= \frac{4h_x^2}{2a_2 \mp a_3h_x - 2a_1h_x^2} && \text{(A.79)}
\end{aligned}$$

When I graph these constraints you can see that they are nearly parallel to the lower poles constraint. With respect to the a_3 parameter, as it increases, the two zeros constraints spread apart. What is interesting is that once the upper zeros constraint begins to intersect the lower poles constraint, the point they intersect approximates where the vertical a_3 constraint appears. The approximation appears to be better as a_3 increases. It is noted in the Observation A.3.1. Figures A.5 and A.6 show the equation graphed with a_3=100 and a_3=1, demonstrating the spreading of the two zero boundary constraints.

Observation A.3.1. *The h_t value where the lower pole constraint and the upper zero boundary constraint intersect appears to converge to the greatest h_t value of the lower boundary for the actual convergence area (the h_t value of the a_3 vertical boundary) as a_3 increases.*

The place where the poles and zeros cross may be causing instability and cause the divergence, however, this is beyond the scope of this thesis and will be left for others to prove. I will use this information to approximate for the a_3 vertical boundary's h_t value. From observations I have made the intersection appears to be a good approximation once the a_3 vertical boundary has receded to a h_t value less than its initial value. This information will be used in Section 5.2.3 on page 43.

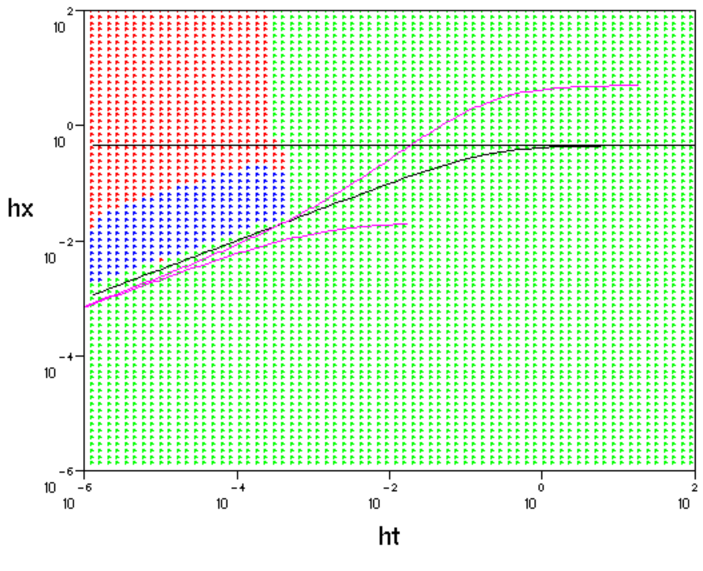

Fig. A.5: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new boundary zeros constraints in magenta at a_3=100.

To use the zero constraint intersection point as an estimation tool, I want to always pick the upper constraint. When the $\mp$ is $-$ and a_3 is positive, it will be the upper

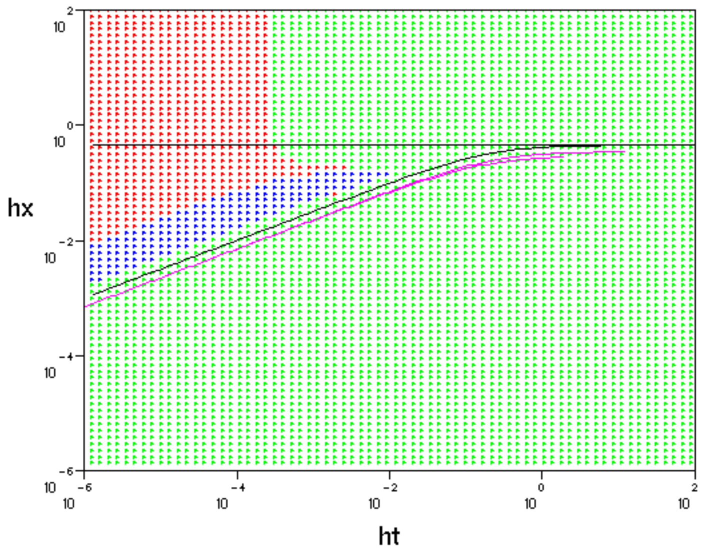

Fig. A.6: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new boundary zeros constraints in magenta at a_3=1.

zeros constraint. When $\mp$ is $+$ and a_3 is negative is the upper limit. In order to make sure I always choose the upper limit, I will make the $\mp$ subtraction and put the absolute value operator around a_3 to get:

$$h_t = \frac{4h_x^2}{2a_2 - |a_3|h_x - 2a_1h_x^2} \qquad (A.80)$$

A.3.3 Graphing the Boundary Zero Constraint When the Right Side is Positive for Backward Euler's

In this section I develop the same equation used in Section A.3.2, the zeros constraint for the boundary values of u, but I will do so for the Backward Euler's Equation. This is because I found that the equation was useful for approximating the a_3 vertical boundary and we now need a version for Backward Euler's.

Some of the algebra steps will be skipped because they are exactly the same equations as the Forward Euler's for the most part, but with a couple extra $\frac{1}{1-a_1h_t}$ terms.

I will begin with an equation similar to Equation A.44 on page 74, but it includes the $\frac{1}{1-a_1h_t}$ terms if Backward Euler's is used instead from Equation 4.26 on page 30. Here $c_1 = \frac{z^{-2}h_t}{(2h_x^2)(1-a_1h_t)\left(1-\frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2}{h_t}+2a_1h_x^2-4a_2\right)(z^{-1})\right)}$:

$$U_j = \frac{\frac{h_t}{2h_x^2(1-a_1h_t)}(c_1(4a_2^2 - a_3^2h_x^2)U_j + 4a_2b_1h_x^2c_1 \pm 2a_3b_1h_x^3c_1 + 2b_1h_x^2)}{\left(1 - \frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1})\right)} \tag{A.81}$$

When I solve for U_j I get:

$$U_j = \frac{\frac{h_t}{2h_x^2(1-a_1h_t)}(4a_2b_1h_x^2c_1 \pm 2a_3b_1h_x^3c_1 + 2b_1h_x^2)}{1 - \frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)(z^{-1}) - \frac{h_t}{2h_x^2(1-a_1h_t)}(c_1)(4a_2^2 - a_3^2h_x^2)} \tag{A.82}$$

Now I substitute c_1 back in and manipulate the terms so that the exponents of z are all positive to find the poles and the zeros of the equation. Also, because the same set of terms are repeated, I am going to let $c_2 = \frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{2h_x^2}{h_t} + 2a_1h_x^2 - 4a_2\right)$. This gives:

$$\begin{aligned} U_j \quad = \quad & \frac{\frac{h_t}{2h_x^2(1-a_1h_t)}(4a_2b_1h_x^2)\left(\frac{z^{-2}h_t}{(2h_x^2)(1-a_1h_t)(1-c_2(z^{-1}))}\right)}{1 - c_2(z^{-1}) - \frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{z^{-2}h_t}{(2h_x^2)(1-a_1h_t)(1-c_2(z^{-1}))}\right)(4a_2^2 - a_3^2h_x^2)} + \\ & \frac{\frac{h_t}{2h_x^2(1-a_1h_t)}(\pm 2a_3b_1h_x^3\left(\frac{z^{-2}h_t}{(2h_x^2)(1-a_1h_t)(1-c_2(z^{-1}))}\right) + 2b_1h_x^2)}{1 - c_2(z^{-1}) - \frac{h_t}{2h_x^2(1-a_1h_t)}\left(\frac{z^{-2}h_t}{(2h_x^2)(1-a_1h_t)(1-c_2(z^{-1}))}\right)(4a_2^2 - a_3^2h_x^2)} \end{aligned} \tag{A.83}$$

After some manipulation I get:

$$U_j \quad = \quad \frac{\frac{b_1h_t}{2h_x^2(1-a_1h_t)^2}\left(2a_2h_t \pm a_3h_xh_t + 2h_x^2(1-a_1h_t)\left(z^2 - c_2z\right)\right)}{\left(z - c_2 - \left(\frac{h_t^2(4a_2^2-a_3^2h_x^2)}{(4h_x^4)(1-a_1h_t)^2(z-c_2)}\right)\right)(z - c_2)} \tag{A.84}$$

To find the zeros constraint I will set the numerator equal to zero and solve for z:

$$0 \quad = \quad \frac{b_1h_t}{2h_x^2(1-a_1h_t)^2}\left(2a_2h_t \pm a_3h_xh_t + 2h_x^2(1-a_1h_t)\left(z^2 - c_2z\right)\right) \tag{A.85}$$

$$= \quad 2a_2h_t \pm a_3h_xh_t + 2h_x^2(1-a_1h_t)\left(z^2 - c_2z\right) \tag{A.86}$$

Using the quadratic equation I get the following constraint by setting the z value to less than 1:

$$1 > \left|\frac{-d_2 \pm \sqrt{d_2^2 - 4d_1 d_3}}{2d_1}\right| \tag{A.87}$$

where

$$\begin{aligned} d_1 &= 2h_x^2 - 2a_1 h_t h_x^2 && \text{(A.88)} \\ d_2 &= -2h_x^2 c_2 + 2a_1 h_t h_x^2 c_2 && \text{(A.89)} \\ d_3 &= 2a_2 h_t \pm a_3 h_x h_t && \text{(A.90)} \end{aligned}$$

Now I will solve for h_t when the right side is negative, which gives the zeros constraints that cross the pole constraints. I first rearrange the equation so that things cancel out before I substitute the d variables back in:

$$\begin{aligned} 1 &= \frac{d_2 \mp \sqrt{d_2^2 - 4d_1 d_3}}{2d_1} && \text{(A.91)} \\ 2d_1 - d_2 &= \mp\sqrt{d_2^2 - 4d_1 d_3} && \text{(A.92)} \\ 4d_1^2 - 4d_1 d_2 + d_2^2 &= d_2^2 - 4d_1 d_3 && \text{(A.93)} \\ 4d_1^2 - 4d_1 d_2 &= -4d_1 d_3 && \text{(A.94)} \\ d_1 - d_2 &= -d_3 && \text{(A.95)} \end{aligned}$$

Now I will substitute back in the d variables and then the c_2 variable. Then I will solve for h_t to graph the equation:

$$\begin{aligned}
2h_x^2 - 2a_1h_th_x^2 + 2h_x^2c_2 - 2a_1h_th_x^2c_2 &= -2a_2h_t \mp a_3h_xh_t & \text{(A.96)}\\
2h_x^2 - 2a_1h_th_x^2 + 2h_x^2(1 - a_1h_t)c_2 &= -2a_2h_t \mp a_3h_xh_t & \text{(A.97)}\\
2h_x^2 - 2a_1h_th_x^2 + 2h_x^2 + 2a_1h_th_x^2 - 4a_2h_t &= -2a_2h_t \mp a_3h_xh_t & \text{(A.98)}\\
4h_x^2 - 2a_2h_t &= \mp a_3h_xh_t & \text{(A.99)}\\
4h_x^2 &= \mp a_3h_xh_t + 2a_2h_t & \text{(A.100)}\\
h_t &= \frac{4h_x^2}{2a_2 \mp a_3h_x} & \text{(A.101)}
\end{aligned}$$

When I graph this equation you can see that, similar to the Forward Euler's zeros constraint, it crosses the lower poles boundary near the a_3 vertical boundary when a_3 is high. This confirms Observation A.3.1 for the Backward Euler's formula. Figures A.7 and A.8 show the zero constraints for the Backward Euler's formula, including the intersection between the zeros constraint and the lower poles constraint in Figure A.7. From the graph I found that when the $\mp$ is $-$ and a_3 is positive, it will be the upper zeros constraint. When $\mp$ is $+$ and a_3 is negative is the upper limit. In order to make sure I always choose the upper limit, I will make the $\mp$ subtraction and put the absolute value operator around a_3 to get:

$$h_t = \frac{4h_x^2}{2a_2 - |a_3|h_x} \qquad \text{(A.102)}$$

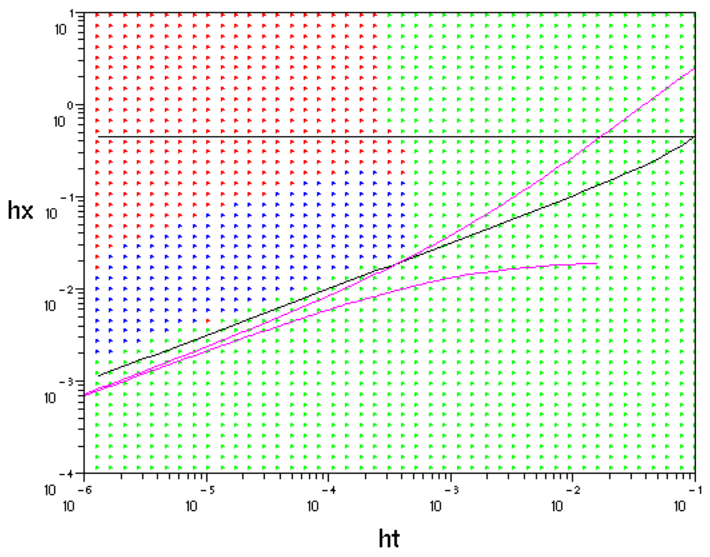

Fig. A.7: Convergence map for the Backward Euler's function showing the original pole constraints in black and the new boundary zeros constraints in magenta at a_3=100.

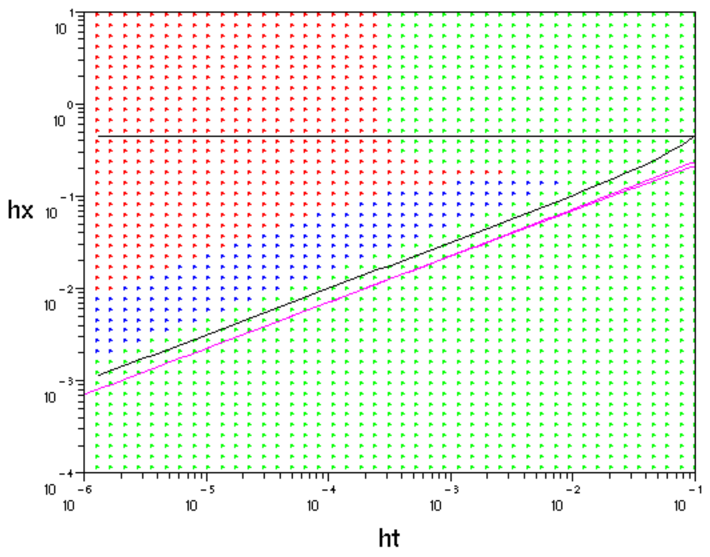

Fig. A.8: Convergence map for the Forward Euler's function showing the original pole constraints in black and the new boundary zeros constraints in magenta at a_3=1.

APPENDIX B

STIFFNESS

In subsection 4.1.1 it was shown that for the general Linear Forward Euler's Equation (Eq. 4.5)

$$\begin{aligned} u_{i+1,j} &= u_{i,j} + h_t \left(a_1 u_{i,j} + b_1 + \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \right. \\ & \left. \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x} \right) \end{aligned} \tag{B.1}$$

that the parameter a_3 creates what appears to be a vertical constraint on stability, a boundary between the areas of convergence and divergence. I tried to explain this in Appendix A on page 67 by examining the zeros constraint further, but this did not yield in any explanations. In this appendix I turned toward the concept of stiffness to see if it could explain this a_3 vertical constraint. From this I was able to actually show the opposite, that stiffness could not be the cause of this behavior for a portion of the convergence area. What follows are the proofs and concepts on the subject of stiffness.

B.1 Stiffness and Eigenvalues

A stiff equation is one where the scaling of terms can cause the equation to become unstable. The stiffness of an equation is quantified best by using eigenvalues. If the equation can be manipulated so that it is in the from of $Ax + c = b$ where x, c, and b are column vectors and A is a square matrix, then the greater difference in the absolute values of the eigenvalues of A, the greater likelihood that the equation will be stiff.

So, to first observe the eigenvalues of our Linear Forward Euler's Equation, I have to manipulate it into the form $Ax + c = b$. To do so let, the number of elements within u be n. The variable i will continue to be the index variable for time and j

the index variable for space, which will be between 1 and n. I can then form the following matrix-vector equation:

$$\begin{bmatrix} u_{i+1,1} \\ u_{i+1,2} \\ u_{i+1,3} \\ \vdots \\ u_{i+1,n-1} \\ u_{i+1,n} \end{bmatrix} = \begin{bmatrix} t_2 & t_3 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ t_1 & t_2 & t_3 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & t_1 & t_2 & t_3 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & t_1 & t_2 & t_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & t_1 & t_2 \end{bmatrix} \begin{bmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \\ \vdots \\ u_{i,n-1} \\ u_{i,n} \end{bmatrix} + \begin{bmatrix} h_t b_1 \\ h_t b_1 \\ h_t b_1 \\ \vdots \\ h_t b_1 \\ h_t b_1 \end{bmatrix} \tag{B.2}$$

where the t variables are defined as:

$$t_1 = h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \tag{B.3}$$

$$t_2 = 1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2} \tag{B.4}$$

$$t_3 = h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \tag{B.5}$$

Although I can't say exactly what the eigenvalues are from our A matrix, I can use the Gershgorin Circle Theorem to find the area they are contained within. The theorem states that for every row, the eigenvalue for the row must lie within a Gershgorin Circle. This circle is on the complex plane, has a center at the diagonal value of the row, and its radius is equal to the absolute value addition of all the other elements of a row. In mathematical terms, the eigenvalue associated for row k lies within the Gershgorin Circle with a center at A_{kk} and has a radius equal to $\sum_{i \neq k} |A_{ki}|$. The following section proves that all of eigenvalues for A are contained within one Gershgorin Circle and how that circle affects the stiffness of the equation as a_3 changes.

B.2 Gershgorin Circles

Lemma B.2.1. *The eigenvalues for the Linear Forward Euler's Equation are contained within the Gershgorin Circle centered at* $1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2}$ *and has a radius of* $\left| h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \right| + \left| h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \right|$.

Proof. From the matrix A that was created in the previous section I have three different distinct rows and therefore three distinct circles:

The first circle is for rows 2 through $n-1$ of A. These rows all have the same diagonal value of $1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2}$, and therefore by definition of the Gershgorin Circle Theorem have a circle centered at the same value. The contents of the non-diagonal entries of A for these rows are the same and when the absolute values of the entries are added up, I get $\left| h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \right| + \left| h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \right|$, which by the theorem becomes the radius of the circle.

The second circle is for row 1 of A, where the diagonal value is also $1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2}$, also has a circle center at the same value by the theorem. The radius of this circle is $\left| h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \right|$. It must therefore be contained or equal to a circle with a radius of $\left| h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \right| + \left| h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \right|$ because the left term is equal to the radius of the circle and the right term will make the radius either exactly the same or larger than the radius of row 1's circle.

The third circle for row n of A also has its diagonal value and center at $1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2}$. The radius of the circle, by the theorem, is $\left| h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \right|$. Using the same argument for row 1's circle, the radius of row n's circle must be contained within the circle with a radius of $\left| h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \right| + \left| h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \right|$, because the radius is equal to the right term and the left term can only be zero or positive.

Therefore, since all of the circles have a center at $1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2}$ and their radii are equal to or less than $\left| h_t \left(\frac{a_2}{h_x^2} - \frac{a_3}{2h_x} \right) \right| + \left| h_t \left(\frac{a_2}{h_x^2} + \frac{a_3}{2h_x} \right) \right|$, all of the eigenvalues of A must be contained within the circle with those values. $\square$

From Lemma B.2.1, I can make an argument that stiffness is not a problem for some areas of convergence. This is possible due to two observations: $a_2 \frac{h_t}{h_x^2}$ along the boundaries of convergence is a constant value and $a_1 h_t$ has a maximum constant value when for both values $a_3 = 0$. Although I cannot prove this is true using theory, the following data tables indicate that these two assertions are true.

The first data tables, Tables B.1 and B.2 show that $a_2 \frac{h_t}{h_x^2}$ is constant along the upper and lower values of convergence (the blue area within the graphs). This comes from the observation that the boundaries appear to follow the equation $h_t = \frac{c}{a_2} h_x^2$ where c is the constant I am going to find. I vary both a_1 and a_2 and find the value of h_x when $h_t = 10^{-5.92}$, the smallest h_t of the data I have collected due to the shape of the convergence area. Both upper and lower boundaries are parallel to each other until h_t becomes to large at a certain point and the upper boundary tapers off and meets the lower boundary. By using the smallest h_t value for the simulations I have run, I will not be calculating $a_2 \frac{h_t}{h_x^2}$ for the tapered off area. Once I have h_x when $h_t = 10^{-5.92}$ for the parameters a_1 and a_2, I then find $a_2 \frac{h_t}{h_x^2}$ to show that their values are extremely close to each other, pointing to that fact that it is most likely a constant value. From the table I will find the average lower boundary constant value is 0.48 and the constant for the upper boundary is about 0.021.

The third data table, Table B.3, points to the fact that $(max(h_t))a_1$ is a constant number, which is easy to see after viewing multiple graphs when varying a_1 and a_2.

a_1	a_2	h_t	h_x	$a_2 \frac{h_t}{h_x^2}$
1	0.01	$10^{-5.92}$	$10^{-3.788}$	$10^{-0.345}$
1	0.1	$10^{-5.92}$	$10^{-3.310}$	$10^{-0.300}$
1	1	$10^{-5.92}$	$10^{-2.780}$	$10^{-0.362}$
1	10	$10^{-5.92}$	$10^{-2.301}$	$10^{-0.317}$
1	100	$10^{-5.92}$	$10^{-1.824}$	$10^{-0.272}$
1	1000	$10^{-5.92}$	$10^{-1.293}$	$10^{-0.334}$
1	10000	$10^{-5.92}$	$10^{-0.815}$	$10^{-0.289}$
0.1	1	$10^{-5.92}$	$10^{-2.780}$	$10^{-0.362}$
10	1	$10^{-5.92}$	$10^{-2.780}$	$10^{-0.362}$
100	1	$10^{-5.92}$	$10^{-2.780}$	$10^{-0.362}$
1000	1	$10^{-5.92}$	$10^{-2.780}$	$10^{-0.362}$

Tab. B.1: Table depicts values of the lower boundary of convergence when $h_t = 10^{-5.92}$, with variations of the parameters a_1 and a_2 varying to compute $a_2 \frac{h_t}{h_x^2}$.

a_1	a_2	h_t	h_x	$a_2\frac{h_t}{h_x^2}$
1	0.01	$10^{-5.92}$	$10^{-3.098}$	$10^{-1.725}$
1	0.1	$10^{-5.92}$	$10^{-2.620}$	$10^{-1.680}$
1	1	$10^{-5.92}$	$10^{-2.142}$	$10^{-1.635}$
1	10	$10^{-5.92}$	$10^{-1.611}$	$10^{-1.697}$
1	100	$10^{-5.92}$	$10^{-1.134}$	$10^{-1.652}$
1	1000	$10^{-5.92}$	$10^{-0.603}$	$10^{-1.714}$
1	10000	$10^{-5.92}$	$10^{-0.125}$	$10^{-1.669}$
0.11	1	$10^{-5.92}$	$10^{-2.142}$	$10^{-1.635}$
10	1	$10^{-5.92}$	$10^{-2.142}$	$10^{-1.635}$
100	1	$10^{-5.92}$	$10^{-2.142}$	$10^{-1.635}$
1000	1	$10^{-5.92}$	$10^{-2.142}$	$10^{-1.635}$

Tab. B.2: Table depicts values of the upper boundary of convergence when $h_t = 10^{-5.92}$, with variations of the parameters a_1 and a_2 varying to compute $a_2\frac{h_t}{h_x^2}$.

In the table I find the largest h_t value and multiply it by a_1 to show that they come to a constant number that is approximately 0.1. This is restated in Observation B.2.2 because it has relevance to estimating the value of the a_3 vertical boundary in Section 5.2.3. This is only an observation because no formal proof is given.

Observation B.2.2. *If $a_3 = 0$, the maximum h_t value for the area of convergence will be approximately $\frac{0.1}{a_1}$.*

a_1	a_2	$max(h_t)$	$(max(h_t))a_1$
0.001	1	$10^{2.009}$	$10^{-0.991}$
0.01	1	$10^{0.980}$	$10^{-1.02}$
0.1	1	$10^{0.0382435}$	$10^{-0.9617565}$
1	1	$10^{-1.012}$	$10^{-1.012}$
10	1	$10^{-1.962}$	$10^{-0.962}$
100	1	$10^{-2.991}$	$10^{-0.991}$
1000	1	$10^{-4.020}$	$10^{-1.020}$
1	0.01	$10^{-1.012}$	$10^{-1.012}$
1	0.1	$10^{-0.932}$	$10^{-0.932}$
1	10	$10^{-0.941}$	$10^{-0.941}$
1	100	$10^{-1.020}$	$10^{-1.02}$
1	1000	$10^{-0.932}$	$10^{-0.932}$

Tab. B.3: Table depicts the maximum h_t value that converges for variations of the parameters a_1 and a_2.

With this information I can find the two circles, one for the lower boundary and one for the upper boundary of convergence. For the lower boundary circle I know that the center is at $1 + h_t a_1 - \frac{2a_2 h_t}{h_x^2}$, where $\frac{2a_2 h_t}{h_x^2} \approx 0.96$ and $0 \leq h_t a_1 \lesssim 0.1$. This means that the center of the circle is $0.04 \lesssim center \lesssim 0.14$. The radius of the circle $2\left|\frac{a_2 h_t}{h_x^2}\right| \approx 0.96$. This means that the smallest eigenvalue possible is -0.92 and the largest possible eigenvalue is 1.1. This gives the maximum difference of eigenvalues, but unfortunately what I am looking for is the difference in absolute values between eigenvalues. Since the smallest absolute value of a possible eigenvalue is zero, and were are comparing degrees of magnitude between eigenvalues, it is possible for stiffness to be a problem for the lower boundary.

However for the upper boundary I can say that in general stiffness is not an issue. Here $\frac{2a_2 h_t}{h_x^2} \approx 0.042$, which gives a center between 0.958 and 1.058. With these circles having a radius of 0.042 the maximum and minimum eigenvalues possible are 0.916 and 1.1. Since the circle's radius does not overlap zero, I can say that a maximum difference in the magnitude of the eigenvalues is 0.184, which is not nearly high enough to cause a stiffness issue.

For the upper boundary I have shown that at $a_3 = 0$ stiffness is not an issue. This continues to be the case as a_3 grows as well, because the factor $\frac{a_3 h_t}{2h_x}$ will grow very slowly because h_t will be very small compared to h_x^2. The parameter a_3 will only be a concern if h_x grows very big to make h_t big. Since this is not the case for the simulations that have been run however, I can say that $\frac{a_3 h_t}{2h_x}$ in general will not contribute to much growth to the radius of the circle for the upper boundary. This means that a_3 is largely independent of the stiffness of the upper boundary and therefore cannot be the cause of the vertical a_3 convergence boundary in the upper area.

I can explicitly define the convergence area where stiffness is not a problem, when the radius of the Gershgorin Circle does not include zero. The smallest the center of a circle can be is $1 - \frac{2a_2h_t}{h_x^2}$, when $h_ta_1 = 0$. When I subtract the radius from this center value I find that stiffness will not be a problem when $a_3 = 0$ for:

$$0 < 1 - \frac{4a_2h_t}{h_x^2} \tag{B.6}$$

$$0.25 < \frac{a_2h_t}{h_x^2} \tag{B.7}$$

The growth of a_3 might be a problem close to this boundary when there is a small margin for error, but as $\frac{a_2h_t}{h_x^2}$ it should not be a problem.

APPENDIX C

SOURCE CODE

This chapter contains the source code that was used to simulate the CA models I created using the Forward and Backward Euler's methods. The code in Section C.1 on page 98 will run multiple CA simulations on a u vector for various h_x and h_t parameters and output the results as text files. The code in Section C.2 on page 104 will take those text files and plot the information to create the convergence maps shown through out this thesis.

C.1 Simulation Code

Listings C.1, C.2, and C.3 are functions used by Listing C.4 to run multiple CA simulations.

Listing C.1: Scilab Function that computes the change in u for a particular cell given its neighbor values for the Forward Euler's function. The function comes from Equation 4.5 on Page 26.

```
//change portion of Forward Euler
//u=current cell value
//uleft and uright are neighbor values
function uChangeAtX=forwardF(u,uleft,uright,ht,hx,a1,a2,a3,b1)
uChangeAtX = (a1*u) + b1;
uChangeAtX = uChangeAtX + ((a2*uright) -
             (2 * a2 * u) + (a2 *uleft))/(hx^2);
uChangeAtX = uChangeAtX + ((a3*uright) - (a3 *uleft))/(2*hx);
uChangeAtX = ht * uChangeAtX;
endfunction
```

Listing C.2: Scilab Function that computes the change in u for a particular cell given its neighbor values for the Backward Euler's function. The function comes from Equation 4.19 on Page 29.

```
//change portion of Backward Euler
//u=current cell value
```

```
//uleft and uright are neighbor values
function uChangeAtX=backF(u,uleft,uright,ht,hx,a1,a2,a3,b1)
//use value from Forward Euler's
uChangeAtX = forwardF(u,uleft,uright,ht,hx,a1,a2,a3,b1);

if(1-(a1*ht)) == 0 then //fix divide by zero error
  uChangeAtX = 10^100;
else
  uChangeAtX = uChangeAtX * (1/(1-(a1*ht)));
end
endfunction
```

Listing C.3: Scilab Function that runs a simulation for u for multiple iterations. The function takes in an initial u vector and outputs the final u vector as well as a value that indicates why the simulation was stopped. See Section 5.1 on page 34 for details about the simulations run.

```
//uSim siulates u function up to maxIter Iterations
//uInit = initial u vector
//form = 1 for Forward Eulers, 2 for Backward Eulers
//uFinal = u vector at end of simulation
//bkReason - reason simulation stopped
//  1=converge, 0=diverge, 7=max iterations
function [uFinal, bkReason]=uSim(uInit,ht,hx,a1,a2,a3,b1,maxIter,form)
bkReason=0;
uFinal = uInit;
[rows,cols] = size(uInit);

//i is time index, j is space index
i=1;
while 1==1 //loop forever, breaks near the end
```

```
i=i+1;

changeU = zeros(rows,1);

for j=1:rows

  if(j-1==0) then

    uLeft = 0; //zero boundary value

  else

    uLeft = uFinal(j-1,i-1);

  end


  if(j+1>rows) then

    uRight = 0; //zero boundary value

  else

    uRight = uFinal(j+1,i-1);

  end


  if(form==1)

   changeU(j,1) = forwardF(uFinal(j,i-1),uLeft,uRight,ht,hx,a1,a2,a3,b1);

  else

   changeU(j,1) = backF(uFinal(j,i-1),uLeft,uRight,ht,hx,a1,a2,a3,b1);

  end


 uFinal(:,i)=changeU + uFinal(:,i-1);


 if(norm(changeU) < 10^(-10)) then

   bkReason=1;

   break;

 end


 if(norm(changeU) > 10^(10)) then
```

```
      bkReason=0;
      break;
    end

    if(i > maxIter) then
      bkReason=7;
      break;
    end
end
endfunction
```

Listing C.4: Scilab script that runs multiple CA simulations on u for different h_x and h_t values. It outputs the simulation results into text files that are used by other scripts to plot the convergence maps.

```
//script that runs a number of ht & hx combos
//and stores the results in files in current working directory
//index.txt - contains results of each ht & hx combo
//ptsConv.txt - list of hx,ht points that converged
//ptsNotConv.txt - list of hx,ht points that diverged
//ptsUnknown.txt - list of hx,ht points that hit max iterations

ptsConv = [0 0];
ptsNotConv = [0 0];
ptsUnknown =[0 0];

//initialize coefficents for and u for simulation
a1=1;
a2=1;
a3=0;
b1=1;
```

```
uInit = [1 2 3 4 5]';

maxIter=4000;


fileNum=1;

numElements = size(uInit,1);

fhandle=file('open','index.txt','unknown');

fprintf(fhandle,'uInit:\n');

for k=1:numElements

   fprintf(fhandle,'%f\n',uInit(k,1)); // write a line

end

fprintf(fhandle,'a1␣=␣%3.15f,␣a2␣=␣%3.15f␣\n',a1,a2); //

fprintf(fhandle,'a3␣=␣%3.15f,␣b1␣=␣%3.15f␣\n\n',a3,b1); //



//for closeup of lower left corner

//generate ht's and hx's to use for simulations

hTExp=[0];

for i=1:75

  hTExp(i) = (10^(-4))*1.2^(i);

  //hTExp(i) = (10^(-3))*1.2^(i);

end

hXExp=[0];

for i=1:75

  hXExp(i) = (10^(-3.5))*1.13^(i);

  //hXExp(i) = (10^(-3))*1.13^(i);

end


//simulate u for each ht,hx pair

for i=1:75
```

```
ht=hTExp(i);
for j=1:75
    hx=hXExp(j);
  if(modulo(j,20) == 0)
    printf('(%d,%d) ',i,j);
  end
  fData = sprintf('data%04d.txt',fileNum);
  [uFinal, bkReason]=uSim(uInit,ht,hx,a1,a2,a3,b1,maxIter,2);

  //uncomment line below to store all u simulation values
  //fprintfMat(fData,uFinal,'%3.15f');
  numIterations = size(uFinal,2);
  fprintf(fhandle,'%s:\n',fData);
  fprintf(fhandle,'ht = %3.15f, hx = %3.15f \n',ht,hx);
  fprintf(fhandle,'Iterations: %d\n',numIterations);
  fprintf(fhandle,'Stable = %d\n',bkReason);
  for k=1:numElements
    fprintf(fhandle,'%3.15f\n',uFinal(k,numIterations));
  end
  fprintf(fhandle,'\n');

  //put hx,ht combo in appropriate list
  if(bkReason == 0)
    ptsNotConv(size(ptsNotConv,1)+1,1) = hx;
    ptsNotConv(size(ptsNotConv,1),2) = ht;
  end

  if(bkReason == 1)
    ptsConv(size(ptsConv,1)+1,1) = hx;
```

```
      ptsConv(size(ptsConv,1),2) = ht;
    end

    if(bkReason == 7)
      ptsUnknown(size(ptsUnknown,1)+1,1) = hx;
      ptsUnknown(size(ptsUnknown,1),2) = ht;
    end

    fileNum = fileNum +1;
  end
end

fprintfMat('ptsNotConv.txt',ptsNotConv(2:size(ptsNotConv,1),:),'%5.18f');
fprintfMat('ptsConv.txt',ptsConv(2:size(ptsConv,1),:),'%5.18f');
fprintfMat('ptsUnknown.txt',ptsUnknown(2:size(ptsUnknown,1),:),'%5.18f');
file('close',fhandle);
```

C.2 Plotting Code

Listings C.5 and C.6 are used by Listing C.7 to plot the convergence maps. Listing C.8 is code that plots the convergence map with the values suggested by the guidelines in Listing 5.1 for the Backward Euler's function.

Listing C.5: Scilab function that returns the h_t value of the zero boundary constraint given the h_x parameter for the Forward Euler's equation. This function comes from Equation A.79 on page 80.

```
function [htPlus,htNeg]=zeroConst1(a1,a2,a3,hx)
  t1 = 4 * hx^2;
  t2 = (-2*a1*hx^2) + (2*a2) +(a3 * hx);
  t3 = (-2*a1*hx^2) + (2*a2) -(a3 * hx);
  htPlus = t1/t2;
  htNeg = t1/t3;
endfunction
```

Listing C.6: Scilab function that returns the h_t value of the zero boundary constraint given the h_x parameter for the Backward Euler's equation. This function comes from Equation A.101 on page 85.

```
function [htPlus,htNeg]=zeroConst1(a1,a2,a3,hx)
  t1 = 4 * hx^2;
  t2 = (2*a2) +(a3 * hx);
  t3 = (2*a2) -(a3 * hx);
  htPlus = t1/t2;
  htNeg = t1/t3;
endfunction
```

Listing C.7: Scilab script used to plot the points in the text files created by Listing C.4. It also plots the poles constraints in black and the zero boundary constraints in magenta. It assumes the text files to plot are in the current working directory.

```
//first graph the simulation data
ptsC = fscanfMat("ptsConv.txt");
ptsN = fscanfMat("ptsNotConv.txt");
ptsU = fscanfMat("ptsUnknown.txt");

ptsCMod = [0];
ptsNMod = [0];
ptsUMod = [0];

set("figure_style","new");
g=gca();
```

```
g.x_label.text = "ht";
g.x_label.font_size = 4;
g.y_label.text = "hx";
g.y_label.font_size = 4;
g.y_label.font_angle = 0;

plot(ptsC(1:size(ptsC,1),2),ptsC(1:size(ptsC,1),1),"b.");
plot(ptsN(1:size(ptsN,1),2),ptsN(1:size(ptsN,1),1),"g.");
plot(ptsU(1:size(ptsU,1),2),ptsU(1:size(ptsU,1),1),"r.");

g.log_flags="ll";
g.children(1).children(1).mark_size = 3;
g.children(2).children(1).mark_size = 3;
g.children(3).children(1).mark_size = 3;

l =legend("Convergent","Divergent","Unknown");

//graph cosntraints
//intialize coefficents
a1=1;
a2=1;
a3=0;

hExp=[0];
for i=1:100
    hExp(i) = (10^(-6))*1.3^(i);
end

//upper poles constraint
```

```
deff("[y]=f(x)","y=sqrt((2*a2)/a1)");
fplot2d(hExp,f);

//Forward Euler's lower poles constraint
//deff("[y]=f(x)","y=sqrt((2*a2*x)/(2+(a1*x)))");

//Backward Euler's lower poles cosnraint
deff("[y]=f(x)","y=sqrt( (2*a2*x)/(2*(1-a1*x)+(a1*x)))");
fplot2d(hExp,f);

//plot zero constraints
hExp2=[0];
for i=1:2000
    hExp2(i) = (10^(-6))*1.05^(i);
end

//zeroConst1 is defined for either Forward or Back Euler's
[htPlus htNeg] =zeroConst1(a1,a2,a3,hExp2(1));
//initialize vectors
zeroConst1PlusCoords = [htPlus hExp2(1)];
zeroConst1NegCoords = [htNeg hExp2(1)];

for i=2:2000
  [htPlus,htNeg] =zeroConst1(a1,a2,a3,hExp2(i));
  zeroConst1PlusCoords(size(zeroConst1PlusCoords,1)+1,:) = [htPlus hExp2(i)];
  zeroConst1NegCoords(size(zeroConst1NegCoords,1)+1,:) = [htNeg hExp2(i)];
end

plot(zeroConst1PlusCoords(:,1),zeroConst1PlusCoords(:,2),"m");
plot(zeroConst1NegCoords(:,1),zeroConst1NegCoords(:,2),"m");

//rezoom
g= gca();
g.zoom_box = [10^(-4),10^(-4),10^(0),10^(0)];
```

Listing C.8: Scilab script used to plot the points in the text files created by Listing C.4. It also plots a magenta line that represents those values recommended by the guidelines in Listing 5.1. It assumes the text files to plot are in the current working directory.

```
//plot simulation data
ptsC = fscanfMat("ptsConv.txt");
ptsN = fscanfMat("ptsNotConv.txt");
ptsU = fscanfMat("ptsUnknown.txt");

ptsCMod = [0];
ptsNMod = [0];
ptsUMod = [0];

set("figure_style","new");
g=gca();

g.x_label.text = "ht";
g.x_label.font_size = 4;
g.y_label.text = "hx";
g.y_label.font_size = 4;
g.y_label.font_angle = 0;

plot(ptsC(1:size(ptsC,1),2),ptsC(1:size(ptsC,1),1),"b.");
plot(ptsN(1:size(ptsN,1),2),ptsN(1:size(ptsN,1),1),"g.");
plot(ptsU(1:size(ptsU,1),2),ptsU(1:size(ptsU,1),1),"r.");

g.log_flags="ll";
g.children(1).children(1).mark_size = 3;
g.children(2).children(1).mark_size = 3;
g.children(3).children(1).mark_size = 3;

a1=10;
a2=1;
a3=100;

//find max ht given
//a1, a2, and a3 coefficents.
upSearch = (0.1/a1)*10^(3);
lowSearch = (0.1/a1)*10^(-15);
multiplier = 1.1;
safetyBuffer = .60;
safetyBuffer2 = 1.6;
htIntersect = -1;
htInit=0.1/a1;

//pick htMax
curHx = lowSearch;
while(curHx < upSearch) //search for intersection
  upperZeroHt = (4*curHx^2)/((2*a2)-(abs(a3)*curHx));
  lowerPoleHt = (2*curHx^2)/((2*a2)+(a1*curHx^2));
  if(upperZeroHt - lowerPoleHt < 0) then
    htIntersect = lowerPoleHt;
    break;
  end
  curHx = curHx * multiplier;
end

if((htIntersect == -1) | (htInit < htIntersect)) then //htInit is lower
  htMax= htInit;
else
  htMax = htIntersect;
end

htMax = htMax * safetyBuffer;

//plot all possible values under htMax from guidelines
hExp=[0];
for i=1:100
    hExp(i) = (10^(-6))*1.3^(i);
end

ht=[0];
hx=[0];
for i=1:100
  if(hExp(i) < htMax) then
    ht(i)=hExp(i);
    hx(i)=safetyBuffer2 *sqrt((2*a2*ht(i))/(2*(1-a1*ht(i))+(a1*ht(i))));
  end
end

plot(ht,hx,"m");
```

REFERENCES

[1] T.A. Burton, editor. *Modeling and Differential Equations in Biology.* Pure and Applied Mathematics. Marcel Dekker Inc., 1980.

[2] J. Curnutt, E. Gomez, and K. E. Schubert. "Patterned Growth in Extreme Environments." 2007.

[3] M. Gardner. "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'." *Scientific American*, (223):120–123, 1970.

[4] R. Hu and X. Ruan. "Differential Equation and Cellular Automata Model." *International Conference on Robotics, Intelligent Systems and Signal Processing*, 2(8-13):1047–1051, October 2003.

[5] D. James. "Cs 322 Project 3: Springies - Backward Euler." http://www.cs.cornell.edu/courses/cs322/2007sp/projects/backwardeuler.pdf, April 2008.

[6] E. Meron, E. Gilad, J. von Hardenberg, M. Shachak, and Y. Zarmi. "Vegetation Patterns Along a Rainfall Gradient." *Chaos, Solitons and Fractals*, 2004.

[7] N. J. Savill and P. Hogeweg. "Competition and Dispersal in Predator-Prey Waves." *Theoretical Population Biology*, (53):243–263, 1999.

[8] K. Schubert. "Cellular Automaton for Bioverms," October 2008.

[9] K. Schubert. *Keith on Numerics.* http://csci.csusb.edu/schubert/pubs/KeithOnNumerical.pdf, Februrary 2008.

[10] J. Shi. "Partial Differential Equations and Mathematical Biology." http://www.resnet.wm.edu/ jxshix/math490/lecture-chap1.pdf, April 2008.

[11] J. Stewart. *Calculus: Concepts and Contexts.* Brooks/Cole Thomson Learning, 2001.

[12] A. M. Turing. "The Chemical Basis of Morphogenesis." *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, August 1952.

[13] J. von Hardenberg, E. Meron, M. Shachak, and Y. Zarmi1. "Diversity of Vegetation Patterns and Desertification." *Physical Review Letters*, 87(19), November 2001.

[14] S. Wolfram. "Twenty Problems in the Theory of Cellular Automata." *Physica Scripta*, T9:170–183, 1985.

[15] S. Wolfram. *A New Kind of Science.* Wolfram Media Inc., 2002.