Simulating Spatial Partial Differential Equations with Cellular Automata

B. Strader[1], **K. Schubert**[1], **E. Gomez**[1], **J. Curnutt**[1], **and P. Boston**[2]
[1]Department of Computer Science and Engineering, California State University, San Bernardino, CA, USA
[2]Department of Earth and Environmental Science, New Mexico Tech, Socorro, NM, USA

Abstract— *Spatial partial differential equations are commonly used to describe systems of biological entities, such as patterns created by desert vegetation and biovermiculation growth, the type of life that could hypothetically live within the caves of Mars. 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 to a local view. This paper discusses how to accomplish this transformation using two methods. The transformation methods are then analyzed using the Z-transform, resulting in guidelines for optimum discretization of space and time in order to achieve convergence and quicker simulations.*

Keywords: cellular automata spatial differential pattern simulation

1. Introduction

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. These equations cover many areas such as population density [5], predatory-prey models [4], morphogenesis [7], and microbial extremophiles [1]. 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. Cellular automata and differential equations have also shown to be related [10] [9].

Approximation techniques are used in this paper to develop direct transformation methods, from partial differential equations to cellular automata models. The rest of the paper includes stability analysis for the new cellular automata models developed, simulations that map out the areas where the models converged to stable values, and guidelines on how to pick the discretization parameters for convergence. For further detail of the methods discussed in this paper see [6].

2. Background

2.1 Defining the Cellular Automata Model

Cellular automata (CA) are simple models that create surprisingly complex results. A CA is composed of a group cells, where the cell grouping can be in different dimensions (vector, matrix, etc). Each cell could contain a variety of values but usually it contains a number or a boolean value. The value of the cell changes each time period of the simulation based upon a set of rules. These rules reference neighboring cell values from the previous time period to calculate current cell values. 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. Over many time periods the values within the cells will usually form some pattern, although it may not be uniform. An example is Conway's "game of life" [2] which simulates how fictitious organisms may live or die within an environment due to their interactions.

Stephen Wolfram, who has researched cellular automata models for several decades, has linked cellular automata with differential equations [9]. In *A New Kind of Science* Wolfram attempts to make a continuous cellular automata model. He allows for an infinite amount of points instead of cells and an infinite amount of gray states between the colors black and white. As a result he found the rule that governs the automata was in fact a differential equation. Wolfram also points out that modified cellular automata are equivalent to Turing machines [10].

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 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 pattern changes as a whole.

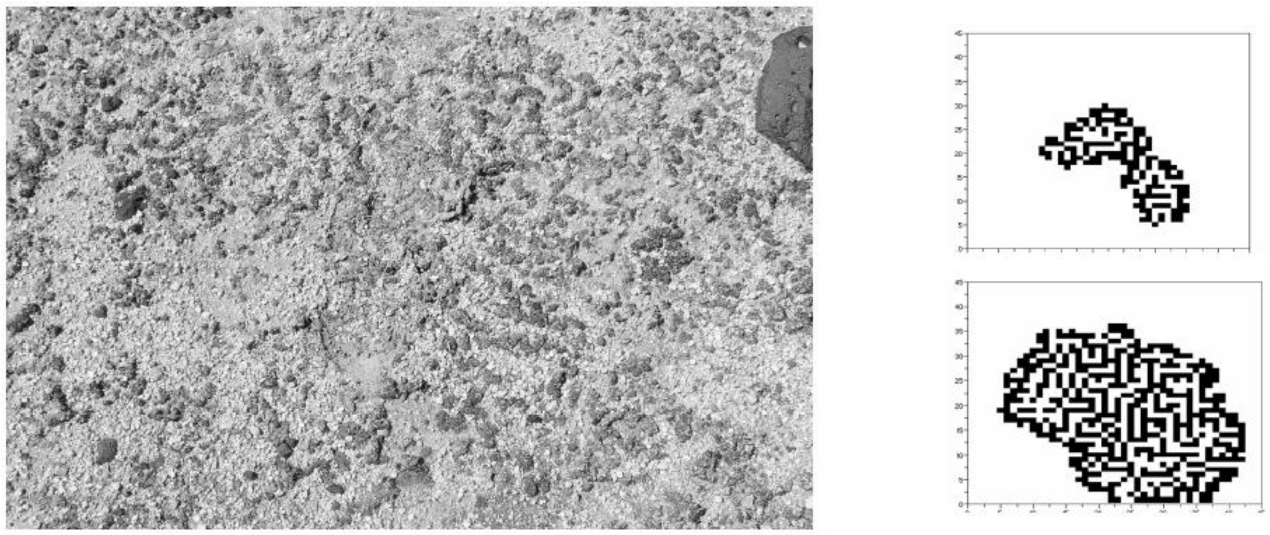

Fig. 1: Images of Cyanobacteria fossils and two patterns created by cellular automata.

2.2 Cellular Automata and Biological Patterns

Cellular automata have been shown to model the growth of real organisms. *[For example, CAs have been recently studied as a model for biovermiculation growth.*

..

Penny's Section Here

..

]

Rules similar to Conway's "game of life" have also been used to model patterns left by Cynobacteria. Figure 1 shows a picture of Cynobacteria fossils on the left and two CA simulations that mimic those patterns [1]. The top simulation in the figure is at an earlier time period of the bottom simulation. The cellular automata model has been recently studied as a model for biological lifeforms such as biovermiculation growth [1]. These lifeforms are extremophiles that receive no sunlight and are of interest because similar lifeforms may be able to live within the caves of Mars.

2.3 Survey of Biological Differential Equations

Traditionally differential equations have been used to describe biological patterns. One example of a biological differential equation that could be simulated by CAs is Fick's Law, which describes population density [5]:

$$\frac{\partial P}{\partial t} = f(t,x,P) + d\nabla_x^2 P \tag{1}$$

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.

Another example are the following equations that deal with dessert vegetation patterns. These are equations that describe plant growth through biomass density $n(x,t)$, and water density $w(x,t)$ [8] [3]:

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

$$\frac{\partial w}{\partial t} = p - (1-\rho n)w - w^2 n + \delta \nabla^2 (w - \beta n) - v\frac{\partial (w - \alpha n)}{\partial x} \tag{3}$$

After a survey of several spatial biological equations, it was found that most can be described by a general differential equation form. Here the variable u is the function for the simulation values with respect to time and one dimensional space, where i is the time index and j is the space index:

$$f(u_{i,j}) = \frac{\partial u}{\partial t} = m(u_{i,j}) + \nabla_x^2 n(u_{i,j}) + \nabla_x o(u_{i,j}) \tag{4}$$

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. Once a differential equation is described in this general form, it can easily be translated to a cellular automata model using the following methods.

3. Conversion to Cellular Automata

Two methods were used to transform Equation 4 into two separate but similar Cellular Automata models. The first method was the Forward Euler's method, used to approximate differential equations:

$$u_{i+1,j} = u_{i,j} + h_t f(u_{i,j}) \tag{5}$$

The variable h_t is defined as the step size between two time values. The function $f(u_{i,j})$ is the differential equation of $u_{i,j}$. Equation 4 can be substituted for $f(u_{i,j})$ but first the equation must be discretized 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 three point formula, which approximates derivatives:

$$g'(x) = \frac{g(x+h) - g(x-h)}{2h} \tag{6}$$

The substitution will create a second step size variable h_x, the step size variable for space. Once the spacial terms are replaced and the differential equation is substituted within the forward Euler's formula the following cellular automata rule is produced:

$$\begin{aligned} u_{i+1,j} \quad = \quad & u_{i,j} + h_t \left(m(u_{i,j}) + \right. \\ & \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \\ & \left. \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right) \end{aligned} \tag{7}$$

From this equation one can construct a cellular automata model. If one lets u be the cells within the cellular automata, then Equation 7 can be used as the simple rule to change cell $u_{i,j}$ from one time period to another. As with other CA models, the equation includes the nearest neighbors of a particular cell ($u_{i,j-1}$ and $u_{i,j+1}$) within the CA rule. This model is mathematically much simpler than the differential equation form (Eq. 4) because the rule only includes the operations addition, subtraction, multiplication, and division.

The second method used was the Backward Euler's method, which is usually more stable than the Forward Euler's Equation for larger values of h_t. The following is the Backward Euler's equation approximated using a first order Taylor series:

$$u_{i+1,j} = u_{i,j} + \frac{h_t f(u_{i,j})}{1 - h_t \frac{\partial f(u)}{\partial u}\Big|_{i,j}} \tag{8}$$

Using similar steps shown above with the Forward Euler's method, the following cellular automata rule is produced:

$$\begin{aligned} u_{i+1,j} \quad = \quad & u_{i,j} + \frac{h_t}{1 - h_t \frac{\partial m(u)}{\partial u}\Big|_{i,j}} \left(m(u_{i,j}) + \right. \\ & \frac{n(u_{i,j+1}) - 2n(u_{i,j}) + n(u_{i,j-1})}{h_x^2} + \\ & \left. \frac{o(u_{i,j+1}) - o(u_{i,j-1})}{2h_x} \right) \end{aligned} \tag{9}$$

This equation can also be used as a CA rule 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, leaving the formula with only addition, subtraction, multiplication, and division. This formula was constructed assuming that $n(u)$ and $o(u)$ contain u variables with a degree of one or less, which usually is the case for the surveyed biological equations.

4. Stability and Z-transform

For the remainder of this paper, the focus will be placed on a particular subset of the general partial differential equation form, one that uses linear terms with respect to u. The following equation was used to represent the general linear form and it can parsed into the format of Equation 4:

$$f(u) = a_1 u + b_1 + \nabla_x^2 (a_2 u) + \nabla_x (a_3 u) \tag{10}$$

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

$$m(u) = a_1 u + b_1 \tag{11}$$

$$n(u) = a_2 u \tag{12}$$

$$o(u) = a_3 u \tag{13}$$

Next the stability of the general formula is analyzed using the Z-transform for both Forward and Backward Euler CA rules. This is done by first manipulating the formula so that the u variables can be transformed easily. The Z-transform is then performed on the CA rule formulas and solved for U_j, the u variable after it has gone through the Z-transform. The poles and zeros for the z variable are then found. 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 resulting inequalities will become constraints on stability. Setting the zeros to less than one are also important for stability and creates additional constraints.

The CA created by the general linear form and the Forward Euler's method is found by substituting equations 11, 12, and 13 into equation 7:

$$\begin{aligned} u_{i+1,j} \quad = \quad & u_{i,j} + h_t \left(a_1 u_{i,j} + b_1 + \right. \\ & \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \\ & \left. \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x} \right) \end{aligned} \tag{14}$$

Similarly the CA rule created by the Backward Euler's equation with the general linear equation is the following:

$$\begin{aligned} &= u_{i,j} + \frac{h_t}{1-a_1 h_t}\Big(a_1 u_{i,j} + b_1 + \\ &\quad \frac{a_2 u_{i,j+1} - 2a_2 u_{i,j} + a_2 u_{i,j-1}}{h_x^2} + \\ &\quad \frac{a_3 u_{i,j+1} - a_3 u_{i,j-1}}{2h_x}\Big) \end{aligned} \tag{15}$$

After the Z-transform is performed and the poles and zeros equations are solved, the constraint created from the zero equations are the same for both Forward and Backward Euler's formulas:

$$\begin{aligned} 1 &> \Big|\frac{-1}{2b_1 h_x^2}((2a_2 - a_3 h_x)(U_{j-1}) + \\ &\quad (2a_2 + a_3 h_x)(U_{j+1}))\Big| \end{aligned} \tag{16}$$

The pole constraint however is different for the two Euler's equations. The Forward Euler's equation produced the the following pole constraint:

$$1 > \left|1 + a_1 h_t - \frac{2a_2 h_t}{h_x^2}\right| \tag{17}$$

The pole constraint for the Backward Euler's equation is:

$$1 > \left|1 + \frac{a_1 h_t}{1 - a_1 h_t} - \frac{2a_2 h_t}{(1 - a_1 h_t) h_x^2}\right| \tag{18}$$

These two formulas show that there needed to be a balancing of h_t and $\frac{h_t}{h_x^2}$ terms 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 one. The third term, however, contains a $\frac{h_t}{h_x^2}$ term, which means that as h_x shrinks, 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 one.

The difference between these two formulas demonstrate why the Backward Euler's formula in general will be more stable than the Forward Euler's formula as h_t becomes 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 for longer periods of time. 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 one. However, as will be discussed in Section 5.3, $a_1 h_t$ at maximum is 0.1 when the CA is stable, meaning both formulas are nearly the same even if h_t becomes large.

5. Convergence Maps and Results

5.1 CA Simulations

In this section there are several graphs depicting convergence maps, using h_t and h_x as parameters. The goal was to discover from these maps the optimal values to pick for the h_t and h_x parameters. The maps were constructed by running either the Forward or Backward Euler's functions (Eqs. 14 and 15) within Scilab for the general linear form, until one of three conditions were met. The first condition was that the values converged, which meant that $\Big|\Big|u_{i+1} - u_i\Big|\Big|_2 < 10^{-10}$. The second was that u was going to diverge, through the condition $\Big|\Big|u_{i+1} - u_i\Big|\Big|_2 > 10^{10}$. The final condition was that the Euler's formula being tested reached the maximum amount of iterations alloted without meeting either the first or second conditions. In the convergence maps that follow, if the Scilab program ended due to convergence, a black dot is placed on the map for corresponding h_t and h_x values used in the program. Similarly, light grey dot was used for h_t and h_x values that diverged, and dark grey 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.

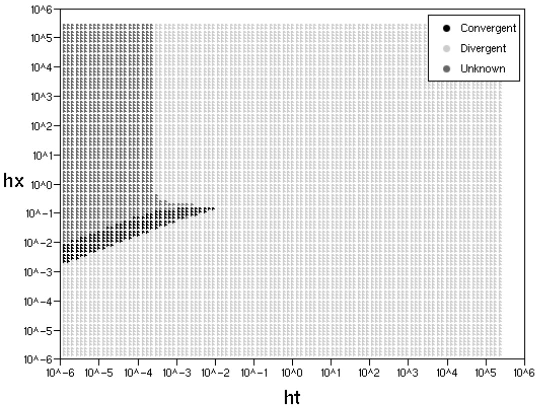

Fig. 2: Convergence map composed of 10,000 simulations with varying h_t and h_x values for the Forward Euler's function.

The first convergence map in Figure 2 is of the Forward Euler's Linear equation, which illustrates how the parameters h_t and h_x affect convergence. Figure 3 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. The values that converged in the upper right hand corner of the convergence map were not deemed as important because they all converged to the same final values.

The lower left area of the convergence map, 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

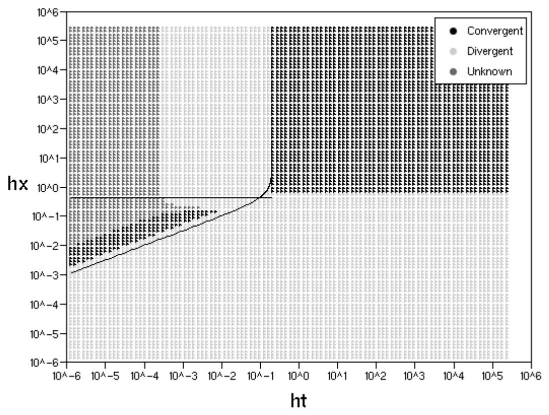

Fig. 3: Convergence map for the Backward Euler's function also containing boundary constraints as black lines.

area 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. The line separating the two halves between convergence and divergence has a slope of approximately $0.48h_x^2$. This is not proven but appears to be the case after being tested for a large number of simulations. The rest of this paper will be focused upon this area of convergence. The values that are converged upon in the area appear to be of importance because 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.

5.2 Graphing Constraints and Iterations

In Figure 3, the pole constraints are graphed onto a convergence map created for the Backward Euler's function. Because the constraint contains an absolute value it leads to two constraints that can be graphed as black lines. Since the Forward Euler's convergence maps are so similar to the Backward Euler's, only the Backward Euler's will be focused upon for the rest of this section. The zero constraint could not be graphed in the form of Equation 16 because it contains the U_{j+1} and U_{j-1} variables, which are not a constant like the a and b variables.

The constraints closely match the area of convergence within the convergence maps, but do not describe it exactly. Through multiple tests, it was observed (but not proven) that the distance between the bottom constraint and the bottom of the convergence area is a constant. The importance of the bottom boundary is found when looking at the number of iterations for each simulation. Those simulations that are closest to the bottom boundary converged the fastest at around 200 iterations and the simulations near the top of the convergence area took about 3000 iterations. This means when picking the h_x and h_t parameters, it is desirable to pick values closest to the bottom constraint for optimal simulation speed.

5.3 Estimating the Third Boundary

The pole constraint uses the a_1 and a_2 values as parameters but not a_3. When a_1 is varied the graph translated diagonally along the slope of the area of convergence. When a_2 is altered the convergence map is translated vertically up and down. Both of these behaviors were predicted by the pole constraints graphed in Figure 3. The b_1 parameter did not appear to affect the area of convergence but instead affected the final convergence values of the CA model. When a_3 is varied, it creates what appears to be a third vertical boundary on the right side of the convergence area. As a_3 increases, the vertical boundary is moved slightly to the right and then crosses back over its initial h_t value and continues toward the left. This is shown in Figure 4.

In order to explain the a_3 parameter, methods were explored to remove the U_{j+1} and U_{j-1} variables from the zeros constraint so it could be graphed, because the formula does contain the a_3 parameter. When the two variables are substituted using a U_j formula that is indexed accordingly, a formula containing U_{j+2} and U_{j-2} is produced. Because these two variables are far enough away from U_j, they were set to zero to produce a graphable constraint. This constraint did not produce any interesting results. However, when the boundary values are considered, u values that do not have either a left or right neighbor, constraints that are slightly different are produced which do appear to have interesting results. Where this new zero boundary constraint crosses the bottom pole constraint is an estimation of where the vertical boundary will be located. This is shown in Figure 5.

This estimation is not very accurate or usable for lower values of a_3, because the new zero boundary constraint will not cross the lower pole constraint at low values of a_3. However, from the observation of multiple simulations, the starting point of the third vertical constraint is a constant value at $\frac{0.1}{a_1}$. The vertical boundary, as mentioned previously, will shift slightly right and then move backward to the left past this initial point. At that time the intersection of constraints becomes accurate enough for estimation purposes. From this information an algorithm to pick near optimal convergence parameters for h_x and h_t was derived.

The algorithm to pick h_x and h_t begins by finding a maximum h_t value. If the intersection between the zero boundary constraint and the lower pole constraint is greater than the initial $\frac{0.1}{a_1}$ value, use $\frac{0.1}{a_1}$ as the maximum h_t value, otherwise use the intersection value. Pick an h_t value lower than the maximum and use that value in the lower pole constraint formula to find an h_x value that is close to the bottom of the convergence area. The h_t and h_x values are also multiplied by some buffer values to ensure a point within the convergence area is picked. This is because the

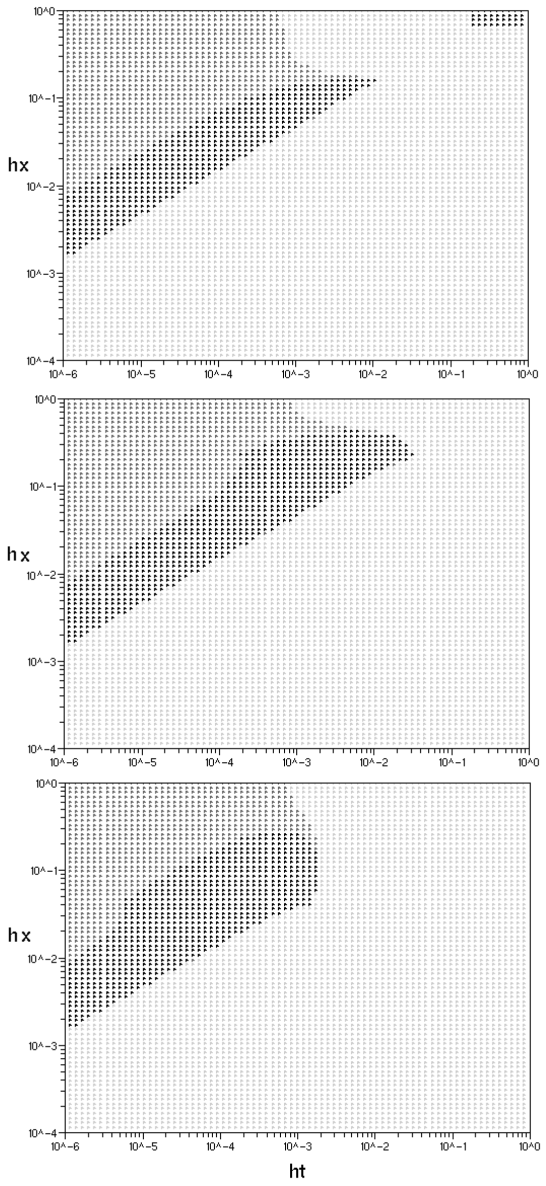

Fig. 4: Convergence maps for the Backward Euler's function demonstrating the a_3 vertical boundary.

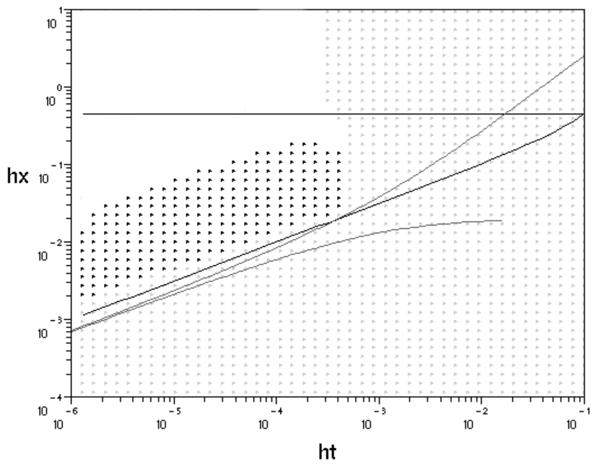

Fig. 5: Convergence map for the Backward Euler's function showing the original pole constraints in black and the new boundary zeros constraints in grey, illustrating the intersection between the two. The unknown points are not shown for clarity.

lower pole constraint is actually below the true area of convergence.

6. Conclusions and Future Work

In conclusion, two methods were found to convert a generalized biological differential equation into a cellular automata model. Both methods produced similar results but both should be continued to be researched for the benefits of each model. A subset formula of the general differential form was then analyzed to find its constraints on stability using the Z-transform. These theoretical constraints matched the shape of the area of convergence in real simulations but did not define them exactly. Multiple tests demonstrated that a third constraint existed that was not predicted by theory, but could be approximated by an intersection of theoretical constraints. This lead to an algorithm that can be used to help a biologist performing CA simulations to pick near optimum step size variables (h_t and h_x) for quicker simulations.

In the future, the model should be analyzed further with regards to stability, allowing for quadratic terms within the formula. Because CA models lend themselves toward parallelism, an efficient means of parallelizing the models proposed should be explored. The third vertical boundary should be explored more fully to see if and how it can be predicated by theory. Hopefully this research will serve as a stepping stone to the ultimate goal of developing a software tool that transform differential equations to cellular automata and preform simulations.

References

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

[2] M. Gardner, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Scientific American*, vol. 223, pp. 120–123, 1970.

[3] E. Meron, E. Gilad, J. von Hardenberg, M. Shachak, and Y. Zarmi, "Vegetation Patterns Along a Rainfall Gradient," *Chaos, Solitons and Fractals*, vol. 19, pp. 367–376, 2004.

[4] N. J. Savill and P. Hogeweg, "Competition and Dispersal in Predator-Prey Waves," *Theoretical Population Biology*, vol. 56, pp. 243–263, 1999.

[5] J. Shi, (2008) Partial Differential Equations and Mathematical Biology. [Online]. Available: http://www.resnet.wm.edu/jxshix/math490/lecture-chap1.pdf

[6] B. Strader, "Simulating Spatial Partial Differential Equations with Cellular Automata," M. CS. thesis, CSU San Bernardino, San Bernardino, CA, USA, Dec. 2008.

[7] A. M. Turing. "The Chemical Basis of Morphogenesis," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 237, no. 641, pp. 37–72, Aug. 1952.

[8] J. von Hardenberg, E. Meron, M. Shachak, and Y. Zarmi1, "Diversity of Vegetation Patterns and Desertification," *Physical Review Letters*, vol. 87, no. 19, pp. 198101-1–198101-4, Nov. 2001.

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

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