4-15-2011

Turing instability in discrete replicator systems

Alex Bryce

Turing Instability in Discrete Replicator Systems

by

Alex J. Bryce

Submitted to the School of Mathematical Sciences
in partial fulfillment of the requirements for the degree of

Masters of Science in Applied & Computational Mathematics

at the

ROCHESTER INSTITUTE OF TECHNOLOGY

April 2011

Author .....
School of Mathematical Sciences
April 15, 2011

Certified by .....
Bernard P. Brooks
Associate Professor
Thesis Supervisor

Accepted by .....
Douglas Meadows
Chairman, School of Mathematical Sciences

Turing Instability in Discrete Replicator Systems

by

Alex J. Bryce

Submitted to the School of Mathematical Sciences
on April 15, 2011, in partial fulfillment of the
requirements for the degree of
Masters of Science in Applied & Computational Mathematics

Abstract

When analyzing a discrete reaction-diffusion dynamical system, one primary area of interest is locating where in the parameter space Turing instabilities occur. It will be shown that Turing instabilities cannot occur in the react then diffuse equations if all diffusion coefficients are equal. The replicator dynamic is a system of equations that is used in evolutionary game theory applications to study behavior types in animal populations. Conditions for a Turing instability in first order discrete replicator systems will be discussed and illustrated with computer simulations of the results.

Thesis Supervisor: Bernard P. Brooks
Title: Associate Professor

Acknowledgments

First and foremost, I'd like to offer my gratitude to my advisor, Dr. Bernard P. Brooks for his consistent patience and support as I worked to complete the thesis while juggling classes, teaching, tutoring, and searching for a job simultaneously. I'd also like to thank Dr. Michael Radin and Dr. Tamas Wiandt for serving on my committee. Their questions and comments forced me to think about ways to write my thesis in order to make it understandable by other readers. Finally, I'd also like to give special thanks to all of the great math interpreters I've had the pleasure to work with at RIT, especially Deborah Makowski who interpreted my meetings with Dr. Brooks for the entire year.

Next, I'd like to thank Dr. Jonathan Forde. My passion for dynamical systems and mathematical biology both stem from the two courses and an independent study I took with him as an undergraduate at Hobart and William Smith Colleges. His enthusiasm for these two subjects sparked my own interest, and inspired me to explore these subjects at RIT. Now that I'm about to receive my graduate degree in applied mathematics, my next goal is to get a doctorate in mathematical biology.

Last, but not the least, I'd like to thank my family and friends. Their patience, love and encouragement helped me to find humor in even the most stressful moments!

Contents

List of Figures

Chapter 1

Introduction

The replicator dynamic has applications in evolutionary game theory, particularly in the analysis of animal behavior. Dawkins defined replicators as entities which can make (in theory) an infinite number of perfect copies of themselves, but their inherent properties can affect their chances of being copied [4]. Hence, sexually reproducing individuals such as human beings are not replicators (since their children are not perfect copies of themselves). However, their genes can be considered to be replicators since genes can be passed on from parent to child indefinitely with an inherent chance of being mutated or not inherited.

For the purposes of this paper, we will focus specifically on behavior types that are passed on from a parent to child within a population of a single generic animal species. Beginning with the discrete replicator dynamic, which describes the changes in the frequency of behavior types over time, we incorporate diffusion into the dynamical system and analyze it for instances of Turing instability.

The paper is divided into two parts. The first half will focus on establishing the theory behind the replicator dynamic and the idea of incorporating diffusion

into a discrete dynamical system. In the second chapter, we will give a formal mathematical definition for the replicator dynamic. In addition, we will establish the framework for applying the replicator system to study the interactions of different behavior tendencies within a population of a single animal species. We also provide 4-dimensional examples. In the third chapter, we will discuss discrete diffusion and temporal and spatial stability, and then we will define Turing instability. We will also construct a general linearized system of equations incorporating diffusion from which we will derive a Jacobian matrix for analyzing the linear spatial stability of equilibrium points of the system.

The second half of the paper will focus on analyzing the replicator system for instances of Turing instability. In the fourth chapter, we will prove that the Jacobian of a replicator system always has a zero determinant. Next, we will prove that it is impossible to cause Turing instability in the two-dimensional replicator system. We will also discuss the possibilities of Turing instability in the three-dimensional and four-dimensional replicator systems. In the fifth chapter, we will consider a similar problem of using diffusion to turn unstable equilibrium points into stable equilibrium points.

Chapter 2

The Discrete Replicator Dynamic

2.1 Defining the Replicator Dynamic

Consider a population consisting of a single animal species with n behavior tendencies. Let $x_i(t)$ be the frequency of type i behavior tendency at discrete time t. Each of these behavior types is considered to be a replicator and we will analyze the fitness of these behaviors in interaction with each other as individuals compete over a needed resource for survival and propagation of the species (such as food or potential mates). Assume that $x_i(t)$ are differentiable functions of time t and that the state of the population at time t is given by the vector $\vec{x}$ which is an element of the unit simplex. That is,

$$\sum_{i=1}^{n} x_i(t) = 1$$

$$0 \leq x_i(t) \leq 1$$

We allow random encounters between individuals. The outcome of these encounters will be determined through a payoff matrix $\mathbf{A}$. The payoff matrix describes the

change in fitness, defined as the number of offspring produced, for the i^{th} type after an encounter with the j^{th} type [8]. We assume without loss of generality that $\mathbf{A}$ is a nonnegative square matrix of size n, which is supported by the interpretation of the interactions: if the i^{th} behavior type individual wins an encounter against the j^{th} behavior type individual, then it is able to produce offspring. However, if the i^{th} behavior type individual loses the encounter, then it is not able to survive and produce offspring. $(\mathbf{A}\vec{x})_i$ is the expected payoff for the type i individual and $\vec{x}^T\mathbf{A}\vec{x}$ is the average payoff for the entire population [5].

We define the general discrete replicator equation as

$$x_i(t+1) = x_i(t)\left(\frac{(\mathbf{A}\vec{x})_i}{\vec{x}^T\mathbf{A}\vec{x}}\right) \tag{2.1}$$

The fraction $\frac{(\mathbf{A}\vec{x})_i}{\vec{x}^T\mathbf{A}\vec{x}}$ compares the fitness of the i^{th} type with the entire population. If the expected payoff for the i^{th} population is greater than the average payoff, then the ratio becomes greater than 1 and the i^{th} population grows. Conversely, if the expected payoff is smaller than the average payoff, then the ratio is smaller than 1 and the i^{th} population shrinks. Since $x_i(t)$ represent population frequencies, they are dimensionless variables. In addition, both the numerator and the denominator of the fraction have the same units (the number of offspring produced) which cancels out. Therefore, the replicator equation is dimensionless.

Since we have n different behavior tendencies, the replicator system is a system of n replicator equations of the form 2.1. Since we are working with a system of difference equations, the equilibrium points are the points such that

$$\vec{x}(t) = f(\vec{x})$$

Note that for any arbitrary i, we have

$$\begin{aligned} x_i(t) &= x_i(t)\left(\frac{(\mathbf{A}\vec{x})_i}{\vec{x}^T\mathbf{A}\vec{x}}\right) \\ 0 &= x_i(t)\left(\frac{(\mathbf{A}\vec{x})_i}{\vec{x}^T\mathbf{A}\vec{x}} - x_i(t)\right) \\ 0 &= x_i(t)\left(\frac{(\mathbf{A}\vec{x})_i}{\vec{x}^T\mathbf{A}\vec{x}} - 1\right) \end{aligned}$$

Since the x_i are constrained on the unit simplex, we ignore the trivial solution: $x_i(t) = 0$ for all i. Note that we can reduce the dimension of the problem by forcing at least one x_i to be zero. If $x_i \neq 0$ for all i, then the equilibrium point is in the interior of the simplex and is the solutions to the following system of linear equations:

$$(\mathbf{A}\vec{x})_1 = (\mathbf{A}\vec{x})_2 = ... = (\mathbf{A}\vec{x})_n$$

$$\sum_{i=1}^{n} x_i(t) = 1$$

As an application of this system, consider a single animal species competing with itself for a food resource. All individuals, upon encountering another individual, can either "display," a method of displaying aggressiveness to scare individuals off without fighting, or "escalate," creating a physical conflict. Within this single species are four behavior tendencies that affect the way an individual interact with another individual when competing over the resource. "Hawks" will always escalate until the opponent flees or either one of them is injured. "Doves" will always display but retreat if the opponent escalates. "Retaliators" will escalate a conflict only if the opponent escalates first. "Bullies" will escalate conflicts, but always retreat if the opponent escalates as well [4].

Assume that when escalating, individuals have a 50% chance of injuring their

opponent and winning the resource and 50% chance of being injured. Assume also that two displaying opponents both have a 50% chance of winning the resource. Let G be the gain in the fitness from winning an encounter and C to be the decrease in fitness from sustaining injuries [1]. We will assume also that $0 < G < C$ [8]. This makes sense as conflict creates a risk of being killed for the combatants (as the loser will be killed or injured during combat, unable to produce offspring). Then we can devise a system of 4 replicator equations with the following fitness matrix:

	Hawk	Dove	Retaliator	Bully
Hawk	$\frac{G-C}{2}$	G	$\frac{G-C}{2}$	G
Dove	0	$\frac{G}{2}$	$\frac{G}{2}$	0
Retaliator	$\frac{G-C}{2}$	$\frac{G}{2}$	$\frac{G}{2}$	G
Bully	0	G	0	0

$$\mathbf{A} = \begin{bmatrix} \frac{G-C}{2} & G & \frac{G-C}{2} & G \\ 0 & \frac{G}{2} & \frac{G}{2} & 0 \\ \frac{G-C}{2} & \frac{G}{2} & \frac{G}{2} & G \\ 0 & G & 0 & 0 \end{bmatrix}$$

Since $\frac{G-C}{2} < 0$ and $\mathbf{A}$ must be nonnegative, we shift $\mathbf{A}$ by adding $\frac{C-G}{2}$ to every entry, giving us the new matrix

$$\mathbf{A} = \begin{bmatrix} 0 & \frac{C+G}{2} & 0 & \frac{C+G}{2} \\ \frac{C-G}{2} & \frac{C}{2} & \frac{C}{2} & \frac{C-G}{2} \\ 0 & \frac{C}{2} & \frac{C}{2} & \frac{C+G}{2} \\ \frac{C-G}{2} & \frac{C+G}{2} & \frac{C-G}{2} & \frac{C-G}{2} \end{bmatrix}$$

This shift has no effect on the behavior of the replicator equation, and so it will not affect the location or the behavior of our equilibrium points [5].

As a simplification of the problem, we proceed to eliminate the parameter G. Since $0 < G < C$, let $G = mC$ where $0 < m < 1$. Making the substitution into the matrix allows us to eliminate G entirely.

$$\mathbf{A} = \begin{bmatrix} 0 & \frac{C+mC}{2} & 0 & \frac{C+mC}{2} \\ \frac{C-mC}{2} & \frac{C}{2} & \frac{C}{2} & \frac{C-mC}{2} \\ 0 & \frac{C}{2} & \frac{C}{2} & \frac{C+mC}{2} \\ \frac{C-mC}{2} & \frac{C+mC}{2} & \frac{C-mC}{2} & \frac{C-mC}{2} \end{bmatrix}$$

A nice feature of this substitution is that everything is written in terms of C, which can be factored out of the matrix as shown:

$$\mathbf{A} = \frac{C}{2} \begin{bmatrix} 0 & 1+m & 0 & 1+m \\ 1-m & 1 & 1 & 1-m \\ 0 & 1 & 1 & 1+m \\ 1-m & 1+m & 1-m & 1-m \end{bmatrix}$$

$$\mathbf{A} = \frac{C}{2}\mathbf{A}_1$$

By factoring out C, we have changed its role in the matrix, from a parameter to a

scalar. We then are able to cancel out C from the replicator equation as follows:

$$x_i(t+1) = x_i(t)\left(\frac{(\mathbf{A}\vec{x})_i}{\vec{x}^T\mathbf{A}\vec{x}}\right)$$
$$x_i(t+1) = x_i(t)\left(\frac{(\frac{C}{2}\mathbf{A}_1\vec{x})_i}{\vec{x}^T\frac{C}{2}\mathbf{A}_1\vec{x}}\right)$$
$$x_i(t+1) = x_i(t)\left(\frac{\frac{C}{2}(\mathbf{A}_1\vec{x})_i}{\frac{C}{2}\vec{x}^T\mathbf{A}_1\vec{x}}\right)$$
$$x_i(t+1) = x_i(t)\left(\frac{(\mathbf{A}_1\vec{x})_i}{\vec{x}^T\mathbf{A}_1\vec{x}}\right)$$

Thus, instead of analyzing the replicator system that contains two parameters C and G, we can analyze the equivalent system that has only one dimensionless parameter $m \in (0, 1)$.

2.2 Equilibirum Points for the Replicator Dynamic

In the two dimensional case (population consists of only Hawks and Doves.), the equilibrium points are as follows:

$$(H, D) \in \{(1, 0), (0, 1), (m, 1-m)\}$$

In the three-dimensional case (population consists of only Hawks, Doves, and Retaliators), the equilibrium points are as follows:

$$(H, D, R) \in \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (m, 1-m, 0), (0, \alpha, 1-\alpha)\}$$

where $0 < \alpha < 1$. In the four-dimensional case (all behavior types are represented), the equilibrium points are as follows:

$$(H, D, R, B) \in \Big\{ (0,0,0,1), (0,0,1,0), (0,1,0,0), (1,0,0,0), (m, 1-m, 0, 0), \\ (0, \alpha, 1-\alpha, 0), \left(\frac{2m}{1+m}, 0, 0, \frac{1-m}{1+m} \right) \Big\}$$

where $0 < \alpha < 1$. The eigenvalues of the Jacobian evaluated at each equilibrium point for each case will be discussed in their respective sections in Chapter 4.

2.3 Replicator Dynamic Examples

As an example, let's investigate how the four-dimensional dynamic behaves if we set $m = 2/3$ and consider different initial population distributions.

Suppose we start with an initial population distribution that is equally divided among four behavior tendencies: $\vec{x} = (0.25, 0.25, 0.25, 0.25)$. Then as time goes by, the interaction between the four behavior types causes the Hawk and the Bully behavior tendencies to become extinct and the population is composed of 80% Retaliators and 20% Doves as shown in Figure 2-1. This corresponds to the equilibirum point $(0, \alpha, 1-\alpha, 0)$ where $\alpha = 0.2$.

Suppose we start with the initial population distribution $\vec{x} = (0.2, 0, 0.1, 0.7)$. Then the dynamic will flow toward a population consisting of only Retaliators as shown in Figure 2-2. This corresponds to the equilibirum point $(0, 0, 1, 0)$.

Finally, suppose we start with the initial population distribution $\vec{x} = (0.8, 0.1, 0.05, 0.05)$. Then the dynamic will flow toward an equilibrium point consisting of 80% Hawks and 20% Bullies as shown in Figure 2-3. This corresponds to the equilibirum point $\left(\frac{2m}{1+m}, 0, 0, \frac{1-m}{1+m}\right)$.

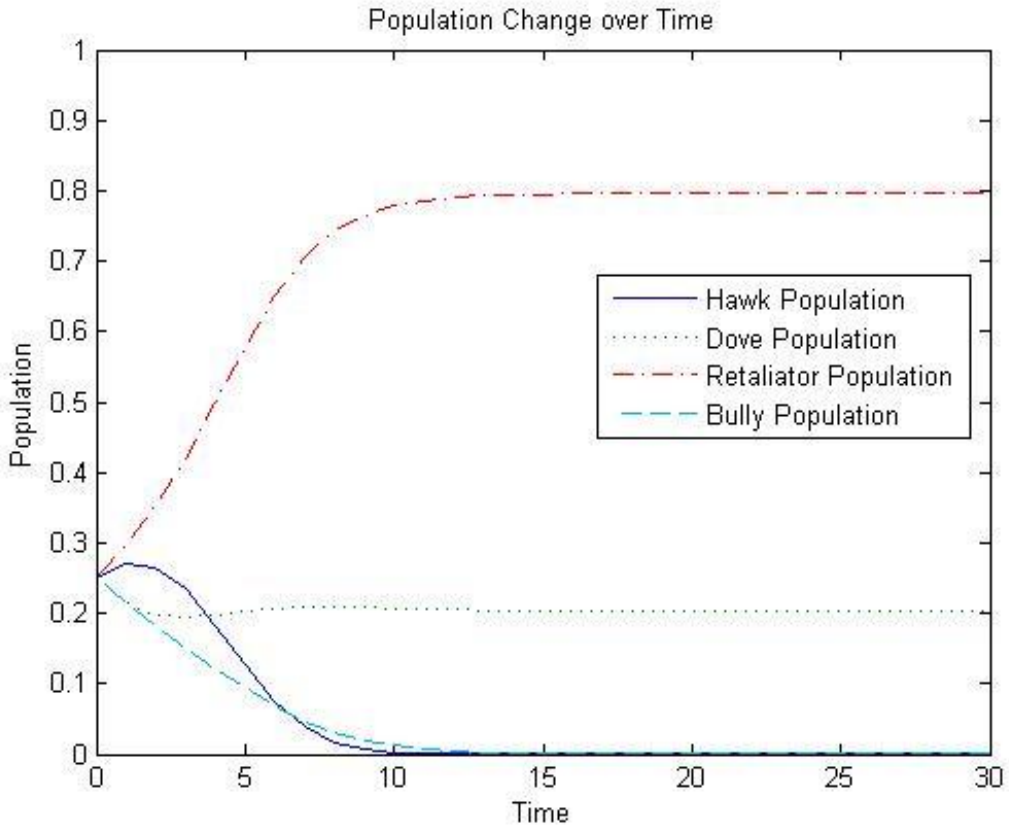


Figure 2-1: Graph showing how the replicator dynamic favors a Retaliator-Dove distribution if we start with equal amounts of all behavior tendencies.

These three figures all show how the replicator dynamic behaves over time for different initial population distributions. Note that we assumed that the entire population resides in a single community where any individual could interact with any other individual. In the next chapter, we will introduce a spatial dimension into the dynamical system, to emulate the concept that individuals may be restricted in whom they are able to interact with because the popluation is distributed among different "communities" and that individuals may try to migrate to different "communities"

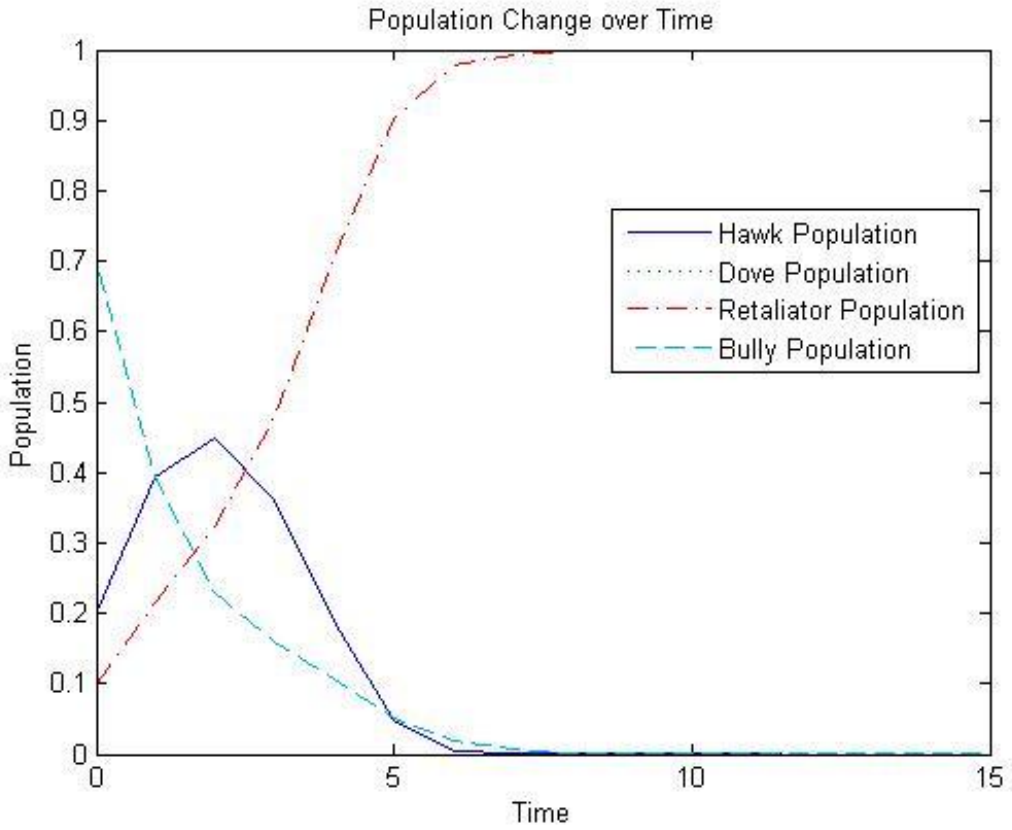


Figure 2-2: Graph showing how using a different initial population distribution causes the replicator dynamic to favor a Retaliator-only distribution.

for better opportunities for survival and propagation of the species.

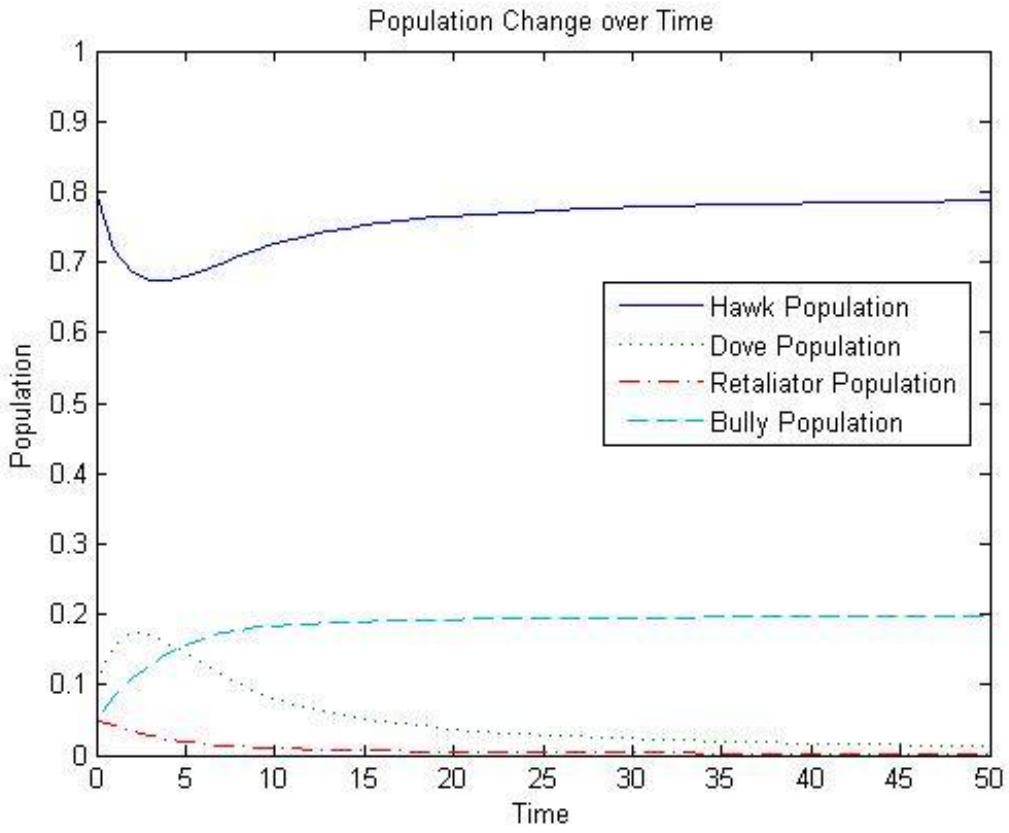


Figure 2-3: Graph showing how using a different initial population distribution causes the replicator dynamic to favor a Hawk-Bully distribution.

Chapter 3

Incorporating Diffusion into the Discrete Dynamical System

3.1 Discussing Discrete Diffusion

When we incorporate diffusion into our dynamical system, we introduce a spatial dimension into our analysis, so that our population dynamic can change through space as well as time. For simpler analysis, we will restrict diffusion to one dimension, so that the population can only move in two directions (i.e., to the left or to the right).

We define our spatial region as a ring of k cells, setting cell 0 and cell k to be the same cell. Let $x_i^s(t)$ be the i^{th} population in cell s at time t. Then for any s such that $1 \leq s \leq k$, populations in cell s can only diffuse with cells $s-1$ and $s+1$. This allows us to avoid dealing with boundary conditions (or to incorporate periodic boundary conditions). With k cells, we now have a system of nk equations.

For each variable $x_i(t)$, there is an associated diffusion constant, $d_i \in [0, 0.5]$. The diffusion constant is the measure of the proportion of the concentration differences

of the population within a cell and its adjacent cells. If we have a full cell and two adjacent cells that are both empty, then we can move up to half of the population from the full cell to each of the empty cells. Hence, the diffusion constant is capped at 0.5.

The ring system has an useful property. Suppose a dynamical system has the equilibrium point $(a_1, a_2, \ldots, a_n)$. When we introduce the ring of spatial cells, we treat each cell as an independent and identical dynamical system, and $(a_1, a_2, \ldots, a_n)$ is also an equilibrium point for each cell. The ring system then has a "global" equilibrium point where $x_i^s = a_i$ for all cells s, $1 \leq s \leq k$ [7].

For example, the Hawk-Dove dynamic has $(m, 1-m)$ as an equilibrium point. If we construct a 3-cell ring system for this dynamic, then each cell can be treated as its own dynamical system that also has $(m, 1-m)$ as an equilibrium point. However, since we must also adhere to the restriction that all population frequencies add up to 1, we have to scale the equilibrium point in each cell. Therefore, the global equilibrium point is $x_1^s = \frac{m}{3}$ and $x_2^s = \frac{1-m}{3}$ for $1 \leq s \leq 3$.

The introduction of diffusion into the dynamical system transforms it from a purely reaction dynamic (where population values changed over time through interaction with other populations) to a reaction-diffusion dynamic. Now population frequencies will be affected by both the interaction with other population frequencies and the spatial motion from one cell to another cell.

In a purely reaction dynamic, the stability of an equilibrium point is purely temporal; either the initial point near the equilibrium point moves away from or closer to the equilibrium point as time progresses. When the spatial dimension is added and the reaction dynamic becomes a reaction-diffusion dynamic, we have to concern ourselves with whether population values in the spatial cells will remain the same or if diffusion will cause the population to migrate to different cells. When

we discuss stability for the reaction-diffusion dynamic, we have to consider both temporal and spatial stability.

An equilibrium point will be linearly temporally stable if all eigenvalues of the Jacobian of the reaction dynamic evaluated at this point have magnitude less than 1. When this occurs, then any initial point near the temporally-stable equilibrium point will move toward it as time passes. If there is at least one eigenvalue that has a magnitude greater than 1, then the equilibrium point will be temporally-unstable and any initial point near the equilibrium point will move away from it as time goes by. If the equilibrium point has at least one eigenvalue equal to 1, then the linear stability test is inconclusive; it may or may not be stable.

An equilibrium point will be linearly spatially stable if, after the ring system and diffusion is incorporated into the reaction dynamical system, all eigenvalues of the Jacobian of the reaction-diffusion dynamic evaluated at this point have magnitude less than 1 for all cells in the system. If there is at least one cell with at least one eigenvalue with a magnitude greater than 1, then the equilibrium point will be spatially unstable. If the equilibrium point has at least one eigenvalue equal to 1 in any cell, then the linear stability test is inconclusive; it may or may not be stable.

Turing instability occurs when a temporally-stable equilibrium point becomes spatially-unstable after diffusion is introduced into the system [5, 7].

As a visual example of a Turing instability, consider the two graphs shown in Figure 3.1, both of which represents the same Dove population, one without diffusion and one with diffusion. In Figure 3-1(a), no diffusion is introduced into the system. Notice that the spatial distribution of the population becomes constant over time. Figure 3-1(b) shows what happens when we introduce diffusion into the system. Notice that the population distribution over the cells are constantly changing. This is the kind of behavior indicative of Turing instability that we are interested in

finding.

The focus of this paper will be to analyze where in parameter space Turing instability occurs in replicator systems. To do so, we need to figure out a way to incorporate diffusion into the dynamical system. Discrete diffusion is more problematic than continuous diffusion. In continuous diffusion, the diffusionable material is able to react with itself and diffuse simultaneously. However, when dealing with discrete diffusion, we have to choose whether to react first then diffuse or to diffuse first then react. In the continuous case, it is impossible to achieve Turing instability if all diffusion coefficients are the same [2]. It will be shown that a similar result occurs for the discrete reaction-diffusion system, making it a better choice over the discrete diffusion-reaction system.

3.2 Constructing the Reaction-Diffusion System

Consider a linearized system of 2 discrete differential equations. Subdivide the spatial region into k cells. Incorporating one-dimensional diffusion into the system (emphasizing that reaction occurs before diffusion) gives us the following system

$$\begin{pmatrix} x^s(t+1) \\ y^s(t+1) \end{pmatrix} = \mathbf{J} \cdot \begin{pmatrix} x^s(t) \\ y^s(t) \end{pmatrix} + \begin{pmatrix} d_x & 0 \\ 0 & d_y \end{pmatrix} \left(\mathbf{J} \cdot \begin{pmatrix} x^{s-1}(t) \\ y^{s-1}(t) \end{pmatrix} - 2\mathbf{J} \cdot \begin{pmatrix} x^s(t) \\ y^s(t) \end{pmatrix} + \mathbf{J} \cdot \begin{pmatrix} x^{s+1}(t) \\ y^{s+1}(t) \end{pmatrix} \right)$$

where $\mathbf{J}$ denotes the Jacobian evaluated at the equilibrium point and d_x and d_y are the diffusion constants.

When we explicitly write out each equation, we obtain $2k$ equations of the form

$$\begin{aligned}
x^s(t+1) &= J_{11}x^s(t) + J_{12}y^s(t) \\
&\quad + d_x J_{11}\left(x^{s-1}(t) - 2x^s(t) + x^{s+1}(t)\right) + d_x J_{12}\left(y^{s-1}(t) - 2y^s(t) + y^{s+1}(t)\right) \\
y^s(t+1) &= J_{21}x^s(t) + J_{22}y^s(t) \\
&\quad + d_y J_{21}\left(x^{s-1}(t) - 2x^s(t) + x^{s+1}(t)\right) + d_x J_{22}\left(y^{s-1}(t) - 2y^s(t) + y^{s+1}(t)\right)
\end{aligned}$$

Notice that both equations depends on the $s-1$, s, and $s+1$ cells. In order to investigate linear stability of the coexistence equilibrium in the presence of diffusion, we will decouple the $2k$ equations by transforming the x and y variables into u and v variables. Define the discrete Fourier transform as

$$\begin{aligned}
x^r(t) = \sum_{r=0}^{k-1} e^{\frac{2irs\pi}{k}} u^r(t) \rightarrow u^r(t) = \frac{1}{k}\sum_{s=0}^{k} e^{\frac{-2irs\pi}{k}} x^s(t) \\
y^r(t) = \sum_{r=0}^{k-1} e^{\frac{2irs\pi}{k}} v^r(t) \rightarrow v^r(t) = \frac{1}{k}\sum_{s=0}^{k} e^{\frac{-2irs\pi}{k}} y^s(t)
\end{aligned}$$

This decoupling transformation will also transform the spatial dimension for easier analysis, so that equations will depend only on cell r instead of cells $s-1$, s, and $s+1$.

Performing the transformation on the first equation:

$$
\begin{aligned}
u^r(t) &= \frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^s(t) \\
&= \frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} \left(J_{11}x^s(t) + J_{12}y^s(t) + d_x J_{11}\left(x^{s-1}(t) - 2x^s(t) + x^{s+1}(t)\right)\right. \\
&\quad \left. + d_y J_{12}\left(y^{s-1}(t) - 2y^s(t) + y^{s+1}(t)\right)\right) \\
&= J_{11}\frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^s(t) + J_{12}\frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} y^s(t) \\
&\quad + d_x J_{11}\left(\frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^{s-1}(t) - \frac{2}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^s(t) + \frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^{s+1}(t)\right) \\
&\quad + d_x J_{12}\left(\frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} y^{s-1}(t) - \frac{2}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} y^s(t) + \frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} y^{s+1}(t)\right) \\
&= J_{11}\frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^s(t) + J_{12}\frac{1}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} y^s(t) \\
&\quad + d_x J_{11}\left(\frac{1}{k}e^{\frac{-2ir\pi}{k}}\sum_{s=1}^{k} e^{\frac{-2ir(s-1)\pi}{k}} x^{s-1}(t) - \frac{2}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} x^s(t)\right. \\
&\quad \left. + \frac{1}{k}e^{\frac{2ir\pi}{k}}\sum_{s=1}^{k} e^{\frac{-2ir(s+1)\pi}{k}} x^{s+1}(t)\right) + d_x J_{12}\left(\frac{1}{k}e^{\frac{-2irs\pi}{k}}\sum_{s=1}^{k} e^{\frac{-2ir(s-1)\pi}{k}} y^{s-1}(t)\right. \\
&\quad \left. - \frac{2}{k}\sum_{s=1}^{k} e^{\frac{-2irs\pi}{k}} y^s(t) + \frac{1}{k}e^{\frac{2ir\pi}{k}}\sum_{s=1}^{k} e^{\frac{-2ir(s+1)\pi}{k}} y^{s+1}(t)\right)
\end{aligned}
$$

Using the fact that if $0 < r < k$, then

$$\sum_{s=1}^{k} e^{\frac{2irs\pi}{k}} = 0$$

and if $r = 0$ or $r = k$, then

$$\sum_{s=1}^{k} e^{\frac{2irs\pi}{k}} = k$$

we simpify the equation to obtain

$$\begin{aligned} u^r(t+1) = J_{11}u^r(t) + J_{12}v^r(t) + d_x J_{11}\left(e^{\frac{-2ir\pi}{k}} - 2 + e^{\frac{2ir\pi}{k}}\right)u^r(t) \\ + d_x J_{12}\left(e^{\frac{-2ir\pi}{k}} - 2 + e^{\frac{2ir\pi}{k}}\right)v^r(t) \end{aligned}$$

Using Euler's formula and trignometric properties, we hide the imaginary terms to get:

$$\begin{aligned} u^r(t+1) &= J_{11}u^r(t) + J_{12}v^r(t) - 4d_x J_{11}\sin^2\left(\frac{r\pi}{k}\right)u^r(t) - 4d_x J_{12}\sin^2\left(\frac{r\pi}{k}\right)v^r(t) \\ &= J_{11}\left(1 - 4d_x\sin^2\left(\frac{r\pi}{k}\right)\right)u^r(t) + J_{12}\left(1 - 4d_x\sin^2\left(\frac{r\pi}{k}\right)\right)v^r(t) \end{aligned}$$

The second equation is transformed similiarly, and we obtain a transformed system of equations:

$$\begin{aligned} u^r(t+1) &= J_{11}\left(1 - 4d_x\sin^2\left(\frac{r\pi}{k}\right)\right)u^r(t) + J_{12}\left(1 - 4d_x\sin^2\left(\frac{r\pi}{k}\right)\right)v^r(t) \\ v^r(t+1) &= J_{21}\left(1 - 4d_y\sin^2\left(\frac{r\pi}{k}\right)\right)u^r(t) + J_{22}\left(1 - 4d_y\sin^2\left(\frac{r\pi}{k}\right)\right)v^r(t) \end{aligned}$$

We can rewrite this linearized system of equations in the following matrix form:

$$\begin{bmatrix} u^r(t+1) \\ v^r(t+1) \end{bmatrix} = \begin{bmatrix} 1 - 4d_x\sin^2\left(\frac{r\pi}{k}\right) & 0 \\ 0 & 1 - 4d_y\sin^2\left(\frac{r\pi}{k}\right) \end{bmatrix} \mathbf{J} \begin{bmatrix} u^r(t) \\ v^r(t) \end{bmatrix}$$

This matrix system of equations represents a linearized two-dimensional discrete dynamical system with diffusion. Since $\mathbf{J}$ is not specified, this system can be applied

to any two-dimensional discrete dynamical system in which we wish to search for Turing instability.

We follow this same procedure for an n-dimensional discrete dynamical system with k cells. After incorporating diffusion into a linearized n-dimensional system, the population vector for cell s, $\overrightarrow{x^s(t)}$, $1 \leq s \leq k$ at any given time t is given by

$$\overrightarrow{x^s(t+1)} = \mathbf{J}\overrightarrow{x^s(t)} + \mathbf{D}\left(\mathbf{J}\overrightarrow{x^{s-1}(t)} - 2\mathbf{J}\overrightarrow{x^s(t)} + \mathbf{J}\overrightarrow{x^{s+1}(t)}\right)$$

where $\mathbf{J}$ denotes the Jacobian of the discrete dynamical system evaluated at the equilibrium point and $\mathbf{D}$ is a diagonal matrix of diffusion constants.

Explictly writing out the equations results in

$$\begin{aligned}
x_1^s(t+1) &= \sum_{j=1}^{n} J_{1j}x_j^s(t) + \sum_{j=1}^{n} d_1 J_{1j}\left(x_j^{s-1}(t) - 2x_j^s(t) + x_j^{s+1}(t)\right) \\
x_2^s(t+1) &= \sum_{j=1}^{n} J_{2j}x_j^s(t) + \sum_{j=1}^{n} d_2 J_{2j}\left(x_j^{s-1}(t) - 2x_j^s(t) + x_j^{s+1}(t)\right) \\
&\vdots \\
x_n^s(t+1) &= \sum_{j=1}^{n} J_{nj}x_j^s(t) + \sum_{j=1}^{n} d_n J_{nj}\left(x_j^{s-1}(t) - 2x_j^s(t) + x_j^{s+1}(t)\right)
\end{aligned}$$

By using the same transformation as presented in the two-dimensional case, we can decouple the equations by a similiar procedure to obtain the following system of

equations:

$$
\begin{aligned}
u_1^r(t+1) &= \sum_{j=1}^{n} J_{1j}\left(1 - 4d_1 \sin\left(\frac{r\pi}{k}\right)\right) u_j^r(t) \\
u_2^r(t+1) &= \sum_{j=1}^{n} J_{2j}\left(1 - 4d_2 \sin\left(\frac{r\pi}{k}\right)\right) u_j^r(t) \\
&\vdots \\
u_n^r(t+1) &= \sum_{j=1}^{n} J_{nj}\left(1 - 4d_n \sin\left(\frac{r\pi}{k}\right)\right) u_j^r(t)
\end{aligned}
$$

which we can rewrite into the following matrix form:

$$
\begin{bmatrix} u_1^r(t+1) \\ \vdots \\ u_n^r(t+1) \end{bmatrix} = \begin{bmatrix} 1 - 4d_x \sin^2\left(\frac{r\pi}{k}\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 - 4d_n \sin^2\left(\frac{r\pi}{k}\right) \end{bmatrix} \mathbf{J} \begin{bmatrix} u_1^r(t) \\ \vdots \\ u_n^r(t) \end{bmatrix} \tag{3.1}
$$

Note that $\mathbf{J}$ is the Jacobian of some given dynamical system. Thus, the given matrix system is a general linearized n-dimensional reaction-diffusion dynamical system that can be applied to any discrete n-dimensional dynamical system in order to search for Turing instability.

The Jacobian of (3.1), which we denote as Γ, is defined as

$$
\Gamma = \begin{bmatrix} 1 - 4d_1 \sin^2\left(\frac{r\pi}{k}\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 - 4d_n \sin^2\left(\frac{r\pi}{k}\right) \end{bmatrix} \mathbf{J} \tag{3.2}
$$

where the trace and the determinant of (3.2) are given as follows:

$$tr(\Gamma) = tr(\mathbf{J}) - 4\sin^2\left(\frac{r\pi}{k}\right)\sum_{i=1}^{n} J_{ii}d_i \tag{3.3}$$

$$det(\Gamma) = det(\mathbf{J})\prod_{i=1}^{n}\left(1 - 4d_i\sin^2\left(\frac{r\pi}{k}\right)\right) \tag{3.4}$$

We can use (3.2) to analyze the stability of equilbirum points of any discrete dynamical system in the presence of diffusion and determine where Turing instabilities occur.

To ensure that our discrete reaction-diffusion dynamical system corresponds with the continuous reaction-diffusion dynamical system, we need to show that Turing instability will not occur if all diffusion coefficients are equal.

Theorem 1. *Turing instabilities cannot occur in a n-dimensional discrete reaction-diffusion dynamical system with one-dimensional diffusion if all diffusion coefficients are equal.*

Proof. Consider a n-dimensional discrete dynamical system with an temporally stable equilbirum point. Let $\mathbf{J}$ be the Jacobian of the dynamical system evaluated at the temporally stable equilibrium point. Let 1-dimensional diffusion be incorporated into the system so it becomes the transformed reaction-diffusion dynamical system given in (3.1).

The Jacobian of (3.1) is given by (3.2). Assume that all diffusion coefficients are equal. That is,

$$0 \leq d_1 = d_2 = \cdots = d_n \leq 0.5$$

Thus Γ changes from the product of two matrices to the product of the Jacobian $\mathbf{J}$

and a scalar:

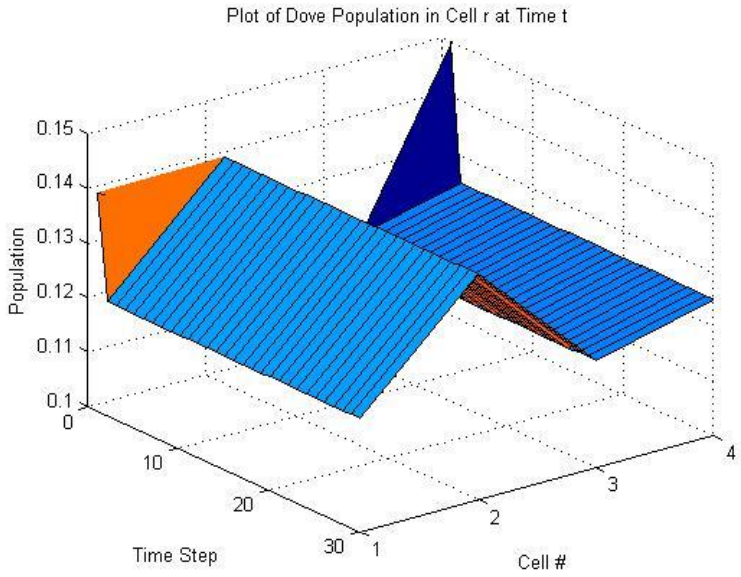
$$\Gamma = \left(1 - 4d_1 \sin^2\left(\frac{r\pi}{k}\right)\right)\mathbf{J}$$

Suppose there is no diffusion, implying that $d_1 = 0$. Then $\Gamma = \mathbf{J}$. Since the equilbirum point is temporally stable by assumption, all eigenvalues of the Jacobian $\mathbf{J}$ have magnitude less than 1, and therefore Γ is also stable.

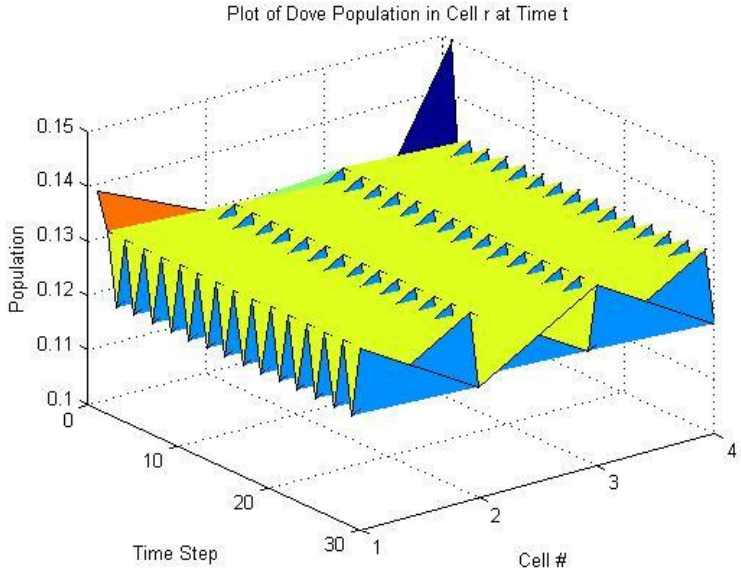
Suppose we maximize diffusion, implying that $d_1 = 0.5$. Then the scalar becomes $\left(1 - 2\sin^2\left(\frac{r\pi}{k}\right)\right)$. Since $1 \leq r \leq k$, $\left(1 - 2\sin^2\left(\frac{r\pi}{k}\right)\right) \in [-1, 1]$. Since $\mathbf{J}$ is evaluated at the temporally-stable equilibrium point, then all of its eigenvalues have magnitude less than 1. Multiplying $\mathbf{J}$ by the scalar shrinks these eigenvalues so the eigenvalues of Γ are guaranteed to have magnitude less than 1. Therefore, the equilibrium point is spatially stable.

Since the temporally-stable equilibrium point is spatially stable for any diffusion constant, Turing instability cannot occur if all diffusion constants are equal. □

Thus, our linearized discrete reaction-diffusion dynamical system corresponds to the continuous reaction-diffusion case, and we can apply this system to the replicator dynamics to search for Turing instability.



(a)



(b)

Figure 3-1: (a) A graph showing what occurs to the Dove population when diffusion is not allowed to occur in the replicator dynamic. (b) A graph showing what happens when diffusion is allowed within the Dove population. This is an example of Turing instability.

Chapter 4

Searching for Turing Instability in Discrete Replicator Systems

4.1 Turing Instability in Replicator Dynamics

Now that we have constructed the general reaction-diffusion dynamical system with one-dimensional diffusion, we can begin to analyze replicator dynamics for presence of Turing instability. First, it is useful to prove one fact about the Jacobian of the replicator system.

Theorem 2. *The determinant of the Jacobian of any n-dimensional replicator system is 0.*

Proof. Consider a n-dimensional replicator system. The variable $x_i(t)$ represents the population frequency of the i^{th} type in the population at time t. By assumption,

$$\sum_{i=1}^{n} x_i(t) = 1$$

Hence, without loss of generality, we see that

$$x_n(t) = 1 - \sum_{i=1}^{n-1} x_i(t)$$
$$x_n(t) = 1 - x_1(t) - x_2(t) - ... - x_{n-1}(t)$$
$$x_n'(t) = -x_1'(t) - x_2'(t) - ... - x_{n-1}'(t)$$

where $x_i(t)$ is the replicator equation for the variable x_i and $'$ represents the partial derivative.

Consider the Jacobian of the replicator system.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x_1(t)}{\partial x_1} & \cdots & \frac{\partial x_1(t)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n(t)}{\partial x_1} & \cdots & \frac{\partial x_n(t)}{\partial x_n} \end{bmatrix}$$

For any arbitrary column j, $\frac{\partial x_n(t)}{\partial x_j}$ can be expressed as a linear combination of the other column entries. That is, for all j, $1 \leq j \leq n$,

$$\frac{\partial x_n(t)}{\partial x_j} = -\sum_{i=1}^{n-1} \frac{\partial x_i(t)}{\partial x_j}$$

Thus, the n^{th} row can be expressed as a linear combination of the other rows. Therefore, $\det(\mathbf{J}) = 0$. □

This fact indicates that for any n-dimensional replicator system, all equilibrium points will have at least one zero eigenvalue.

4.2 Turing Instability in the Two-Dimensional Case

Using the fact that the determinant of the Jacobian is always zero, we can prove that Turing instability cannot occur in the two-dimensional case.

Theorem 3. *Turing instability cannot occur in the two-dimensional case.*

Proof. Consider the general two-dimensional discrete replicator system with non-overlapping generations where $x_1(t)$ and $x_2(t)$ are nonnegative for all t. Such a system has the following form:

$$x_1(t+1) = x_1(t)\left(\frac{(\mathbf{A}\vec{x})_1}{\vec{x}^T\mathbf{A}\vec{x}}\right)$$
$$x_2(t+1) = x_2(t)\left(\frac{(\mathbf{A}\vec{x})_2}{\vec{x}^T\mathbf{A}\vec{x}}\right)$$

where $\mathbf{A}$ is a general nonnegative 2 by 2 matrix. Within the evolutionary game theory framework, we can perceive this as a system describing the dynamic between the Hawk ($x_1(t)$) population and the Dove ($x_2(t)$) population where their interaction is determined by the following matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 1+m \\ 1-m & 1 \end{bmatrix}$$

However, for the proof, we will consider a general nonnegative matrix $\mathbf{A}$.

The Jacobian, $\mathbf{J}$ of the system evaluated at a given point (x_1, x_2) is given by

$$\mathbf{J} = \begin{bmatrix} \frac{a_{11}a_{21}x_1^2x_2+2a_{11}a_{22}x_1x_2^2+a_{12}a_{22}x_2^3}{(\vec{x}^T\mathbf{A}\vec{x})^2} & \frac{-a_{11}a_{21}x_1^3-2a_{11}a_{22}x_1^2x_2-a_{12}a_{22}x_1x_2^2}{(\vec{x}^T\mathbf{A}\vec{x})^2} \\ \frac{-a_{11}a_{21}x_1^2x_2-2a_{11}a_{22}x_1x_2^2-a_{12}a_{22}x_2^3}{(\vec{x}^T\mathbf{A}\vec{x})^2} & \frac{a_{11}a_{21}x_1^3+2a_{11}a_{22}x_1^2x_2+a_{12}a_{22}x_1x_2^2}{(\vec{x}^T\mathbf{A}\vec{x})^2} \end{bmatrix}$$

Since $\vec{x}$ and $\mathbf{A}$ are both nonegative, $tr(\mathbf{J}) \geq 0$.

Suppose we construct a spatial region of k cells and incorporate diffusion into the system. The Jacobian of the two-dimensional reaction-diffusion system, Γ, is given by (3.2). We avoid Turing instability if the eigenvalues of Γ have magnitude less than 1, which will occur if the following inequalities are satisfied [6]:

$$\begin{aligned} det(\Gamma) &< 1 \\ -tr(\Gamma - 1 &< det(\Gamma) \\ tr(\Gamma - 1 &< det(\Gamma) \end{aligned}$$

where, according to (3.4) and (3.3),

$$\begin{aligned} det(\Gamma) &= det(\mathbf{J}) \left(1 - 4d_1 \sin^2\left(\frac{r\pi}{k}\right)\right) \left(1 - 4d_2 \sin^2\left(\frac{r\pi}{k}\right)\right) \\ tr(\Gamma &= tr(\mathbf{J}) - 4\sin^2\left(\frac{r\pi}{k}\right)(d_1 J_{11} + d_2 J_{22}) \end{aligned}$$

where r denotes the r^{th} cell, $1 \leq r \leq k$.

Since $det(\mathbf{J}) = 0$, then $det(\Gamma) = 0 < 1$. Thus, we only need to show that $-1 < tr(\Gamma) < 1$.

Suppose (x_1, x_2) is a temporally-stable equilibrium point of the system. Then the Jacobian, $\mathbf{J}$, evaluated at the point (x_1, x_2) satisfies the following inequalities [6]:

$$\begin{aligned} det(\mathbf{J}) &< 1 \\ -tr(\mathbf{J}) - 1 &< det(\mathbf{J}) \\ tr(\mathbf{J}) - 1 &< det(\mathbf{J}) \end{aligned}$$

Since $det(\mathbf{J}) = 0$ and $tr(\mathbf{J}) \geq 0$, we have $0 \leq tr(\mathbf{J}) < 1$. Assume without loss of

generality that $0 \leq d_1 \leq d_2 \leq 0.5$. Then it follows that

$$\begin{aligned} J_{11}d_1 + J_{22}d_2 &\leq J_{11}d_2 + J_{22}d_2 \\ &\leq tr(\mathbf{J})d_2 \\ &\leq \frac{tr(\mathbf{J}}{2} \\ &< \frac{1}{2} \end{aligned}$$

Thus, we know that the term $4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2) \in (0, 2)$. We sum the inequalities

$$\begin{aligned} 0 < &\quad tr(\mathbf{J}) &< 1 \\ -2 < &\quad -4\sin^2\left(\tfrac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2) &< 0 \end{aligned}$$

to obtain

$$-2 < tr(\mathbf{J} - 4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2) < 1$$

Hence, $tr(\Gamma) < 1$.

To show that $-1 < tr(\Gamma)$, recall that without loss of generality,

$$J_{11}d_1 + J_{22}d_2 \leq tr(\mathbf{J})d_2$$

Suppose $d_1 \neq d_2$. Then this is a strict inequality. Therefore,

$$\begin{aligned}
4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2) &< 4\sin^2\left(\frac{r\pi}{k}\right)d_2 tr(\mathbf{J}) \\
-4\sin^2\left(\frac{r\pi}{k}\right)d_2 tr(\mathbf{J}) &< -4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2) \\
tr(\mathbf{J}) - 4\sin^2\left(\frac{r\pi}{k}\right)d_2 tr(\mathbf{J}) &< tr(\mathbf{J}) - 4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2) \\
tr(\mathbf{J})\left(1 - 4\sin^2\left(\frac{r\pi}{k}d_2\right)\right) &< tr(\mathbf{J}) - 4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2)
\end{aligned}$$

Note that since $tr(\mathbf{J}) < 1$ and $d_2 \in [0, 0.5]$. Then we see that

$$-1 < tr(\mathbf{J})\left(1 - 4\sin^2\left(\frac{r\pi}{k}d_2\right)\right) < 1$$

and so we have

$$1 < tr(\mathbf{J})\left(1 - 4\sin^2\left(\frac{r\pi}{k}d_2\right)\right) < tr(\mathbf{J}) - 4\sin^2\left(\frac{r\pi}{k}\right)(J_{11}d_1 + J_{22}d_2)$$

which shows $tr(\Gamma) > -1$ as desired.

Since $det(\Gamma) = 0$ and $|tr(\Gamma)| < 1$, Γ satisfies the three inequalities necessary for stability. Thus, it is impossible to achieve Turing instability in the 2-dimenisonal case. □

4.3 Turing Instability in the Three-Dimensional Case

Having proven that Turing instability cannot occur in the two dimensional case, we turn our attention to the 3-dimensional case. That is, we now consider a discrete

replicator system consisting of 3 equations:

$$x_1(t+1) = x_1(t)\left(\frac{(\mathbf{A}\vec{x})_1}{\vec{x}^T\mathbf{A}\vec{x}}\right)$$
$$x_2(t+1) = x_2(t)\left(\frac{(\mathbf{A}\vec{x})_2}{\vec{x}^T\mathbf{A}\vec{x}}\right)$$
$$x_3(t+1) = x_3(t)\left(\frac{(\mathbf{A}\vec{x})_3}{\vec{x}^T\mathbf{A}\vec{x}}\right)$$

where (A) is the fitness matrix

$$\begin{bmatrix} 0 & 1+m & 0 \\ 1-m & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

with $0 < m < 1$. This system describes the dynamic between three behavior tendencies: Hawks ($x_1(t)$), Doves ($x_2(t)$), and Retaliators ($x_3(t)$).

The equilibirum points of the system and their corresponding eigenvalues are given in a table below:

Equilibrium Point	Eigenvalues	Linear Stability Test
$(1, 0, 0)$	Singularity (Division by 0)	Numerical results suggest instability
$(0, 1, 0)$	0, $1+m$, 1	Unstable Equilbirum Point
$(0, 0, 1)$	0, 0, 1	Inconclusive by the Linear Stability Test
$(m, 1-m, 0)$	0, $\frac{1}{1+m}$, $\frac{1}{1+m}$	Stable Equilibrium Point
$(0, \alpha, 1-\alpha)$ $0 < \alpha < 1$	0, $\alpha(1+m)$, 1	Unstable Equilbirum Point if $\frac{1}{1+m} < \alpha < 1$

Thus, $(m, 1-m, 0)$ is the only linear temporally stable equilbirum point which we need to analyze for Turing instabilty. The equilbirum point $(0, \alpha, 1-\alpha)$ is a special case which will be discussed separately.

Introducing diffusion to the three-dimensional system results in a new Jacobian of the form (3.2). By Theorem 2, $det(\mathbf{J}) = 0$. This together with (3.4) implies $det(\Gamma) = 0$. By [3], the presence of Turing instability will depend on the trace and the principal minors of Γ. Starting with a temporally stable equilbirum point, Γ will be spatially stable if the following inequalities are satisfied:

$$\sum_{i=1}^{3} M_i(\Gamma) < 1$$

$$tr((\Gamma) - 1 < \sum_{i=1}^{3} M_i(\Gamma)$$

$$-tr((\Gamma) - 1 < \sum_{i=1}^{3} M_i(\Gamma)$$

The analysis becomes much harder in this case. Previously, in the two-dimensional case, we only had to concern ourselves with the trace of $\mathbf{J}$ and how it relates to trace of Γ. Now, in order to look for Turing instability, we have to consider the relationship between the trace and principal minors of $\mathbf{J}$ and the trace and the principal minors of Γ.

We will resort to using numerical simulations to explore the parameter space and locate where Turing instabilities occur. Since Turing instability does not occur when there is no diffusion in the system, intuition suggests that if Turing instability cannot occur even if diffusion is maximized, then it is not possible to cause Turing instability in the system.

The Jacobian of the replicator system evaluated at the equilibrium point

$(m, 1-m, 0)$ is given as follows:

$$\mathbf{J} = \begin{bmatrix} \frac{1-m}{1+m} & \frac{-m}{1+m} & \frac{-2m}{1+m} \\ \frac{m-1}{1+m} & \frac{m}{1+m} & \frac{2m-1}{1+m} \\ 0 & 0 & \frac{1}{1+m} \end{bmatrix}$$

From (3.2), after incorporating diffusion into the system, the new Jacobian has the form

$$\Gamma = \begin{bmatrix} 1 - 4d_1 \sin^2\left(\frac{r\pi}{k}\right) & 0 & 0 \\ 0 & 1 - 4d_2 \sin^2\left(\frac{r\pi}{k}\right) & 0 \\ 0 & 0 & 1 - 4d_3 \sin^2\left(\frac{r\pi}{k}\right) \end{bmatrix} \mathbf{J}$$

where k is the number of spatial cells introduced into the system and $d_i \in [0, 0.5]$. We want to try maximizing the magnitude of the diagonal entries in the diffusion matrix. The term $\sin^2\left(\frac{r\pi}{k}\right)$ achieves its maximum value of 1 when the fraction $\frac{r}{k}$ is exactly $\frac{1}{2}$. That can only occur when we have an even number of cells, which we will assume. When that occurs, each diagonal entry $D_i = 1 - 4d_i \sin^2\left(\frac{r\pi}{k}\right) \in [-1, 1]$ for $i = 1, 2, 3$.

We will consider all possible (D_1, D_2, D_3) tuples where $D_i \in -1, 0, 1$ and see if there exist some tuple that will cause Turing instability by breaking one of the inequalities necessary for stability. There are 27 possible tuples. By Theorem 1, we disregard the $(0, 0, 0)$, $(-1, -1, -1)$, and $(1, 1, 1)$ tuples since they correspond to the case where all diffusion coefficients are equal. Thus, we have twenty-four cases we need to check.

We multiply the two matrices

$$\Gamma = \begin{bmatrix} 1 - 4d_1 \sin^2\left(\frac{r\pi}{k}\right) & 0 & 0 \\ 0 & 1 - 4d_2 \sin^2\left(\frac{r\pi}{k}\right) & 0 \\ 0 & 0 & 1 - 4d_3 \sin^2\left(\frac{r\pi}{k}\right) \end{bmatrix} \begin{bmatrix} \frac{1-m}{1+m} & \frac{-m}{1+m} & \frac{-2m}{1+m} \\ \frac{m-1}{1+m} & \frac{m}{1+m} & \frac{2m-1}{1+m} \\ 0 & 0 & \frac{1}{1+m} \end{bmatrix}$$

to obtain

$$\Gamma = \begin{bmatrix} \frac{D_1(1-m)}{1+m} & \frac{-D_1 m}{1+m} & \frac{-2D_1 m}{1+m} \\ \frac{D_2(m-1)}{1+m} & \frac{D_2 m}{1+m} & \frac{D_2(2m-1)}{1+m} \\ 0 & 0 & \frac{D_3}{1+m} \end{bmatrix}$$

The trace and the principal minors of Γ are

$$tr(\Gamma) = \frac{D_3 + D_2 m + D_1 - D_1 m}{1+m}$$
$$\sum_{i=1}^{3} M_i(\Gamma) = \frac{D_3(D_1 + D_2 m - D_1 m)}{(1+m)^2}$$

We plug each (D_1, D_2, D_3) tuple into the above equations and then vary m to see whether the linear stability conditions are broken. That is, to avoid Turing instabilty, we want to verify that the inequalities

$$D_3(D_1 + D_2 m - D_1 m) < (1+m)^2$$
$$\frac{\pm(D_3 + D_2 m + D_1 - D_1 m) - 1 - m}{1+m} < \frac{D_3(D_1 + D_2 m - D_1 m)}{(1+m)^2}$$

are not violated for $0 < m < 1$. We provide three examples.

Example 1: $(0, 0, 1)$

$$\begin{aligned} 1(0 + m(0 - 0)) = \quad 0 \quad &< (1 + m)^2 \\ \frac{(1 + 0 + m(0 - 0)) - 1 - m}{1 + m} = \quad \frac{-m}{1+m} \quad &< 0 \\ \frac{-(1 + 0 + m(0 - 0)) - 1 - m}{1 + m} = \quad \frac{-2-m}{1+m} \quad &< 0 \end{aligned}$$

The inequalities holds for all $0 < m < 1$, so the $(0, 0, 1)$ tuple will not cause Turing instability.

Example 2: $(1, 1, -1)$

$$\begin{aligned} (-1)(1 + (1)m - (1)m) < \quad (1 + m)^2 \quad &\rightarrow -1 < (1 + m)^2 \\ \frac{(-1 + (1)m + 1 - (1)m) - 1 - m}{1 + m} < \quad \frac{(-1)(1+(1)m-(1)m)}{(1+m)^2} \quad &\rightarrow -1 < \frac{-1}{(1 + m)^2} \\ \frac{-(-1 + (1)m + 1 - (1)m) - 1 - m}{1 + m} < \quad \frac{(-1)(1+(1)m-(1)m)}{(1+m)^2} \quad &\rightarrow -1 < \frac{-1}{(1 + m)^2} \end{aligned}$$

The inequalities holds for all $0 < m < 1$, so the $(1, 1, -1)$ tuple will not cause Turing instability.

Example 3: $(1, 0, -1)$

$$\begin{aligned} (-1)(1 + (0)m - (1)m) < \quad (1 + m)^2 \quad &\rightarrow m - 1 < (1 + m)^2 \\ \frac{((-1) + (0)m + 1 - (1)m) - 1 - m}{1 + m} < \quad \frac{(-1)(1+(0)m-(1)m)}{(1+m)^2} \quad &\rightarrow \frac{-1 - 2m}{1 + m} < \frac{m - 1}{(1 + m)^2} \\ \frac{-((-1) + (0)m + 1 - (1)m) - 1 - m}{1 + m} < \quad \frac{(-1)(1+(0)m-(1)m)}{(1+m)^2} \quad &\rightarrow \frac{-1}{1 + m} < \frac{m - 1}{(1 + m)^2} \end{aligned}$$

The inequalities holds for all $0 < m < 1$, so the $(1, 0, -1)$ tuple will not cause Turing instability.

Following this type of analysis for each (D_1, D_2, D_3) tuple reveals that the condi-

tions for stability are never broken, indicating that the temporally stable equilibrium point $(m, 1-m, 0)$ is always spatially stable and Turing instability is not possible.

As a sidenote, consider the equilibrium point $(0, \alpha, 1-\alpha)$. Since it has an eigenvalue of 1, it fails the linear stability test and we cannot be sure that it is stable or unstable. Consider the example where $\alpha = m = 0.5$ and we have a spatial region of six cells. Since $\alpha = 0.5$, then the equilibirum point is $\left(0, \frac{1}{2}, \frac{1}{2}\right)$. With six spatial cells, this translates to each cell having the following population distribution: 0 Hawks, $\frac{1}{12}$ Doves, and $\frac{1}{12}$ Retaliators.

If we introduce no diffusion in the system, we get the behavior that we expect from the equilibirum point: for all time t, in all cells r, the population distribution is constant, unchanged from the initial population distribution of 0 Hawks, $\frac{1}{12}$ Doves, and $\frac{1}{12}$ Retaliators in each cell. We perturb the initial population distribution by altering only the values of the Doves and Retaliators in each cell. The dynamic using this perturbed initial distribution reaches spatial equilibrium as indicated by Figure 4-1(a) and Figure 4-2(a). In both figures, the Dove and the Retaliator populations in each cell becomes stable over time.

Without diffusion, the system reaches a stable spatial distribution of the population. Once we introduce diffusion into the system, the system is not able to reach a spatial equilibirum as shown in Figure 4-1(b) and Figure 4-2(b). Both the Dove and the Retaliator population in each cell oscillates between 0.0832 and 0.0835 indefinitely. This is an example of spatial instability.

Recall that Turing instability occurs when a temporally stable equilibrium becomes spatially unstable. Since $(0, \alpha, 1-\alpha)$ has an eigenvalue of 1, it fails the linear stability test and we cannot be certain that this equilibrium point is linear temporally stable or not. Hence, we cannot use this as an example of Turing instability because of our strict definition. However, this is precisely the type of behavior that

we are looking for when searching for Turing instability.

4.4 Turing Instability in the Four-Dimensional Case

Consider the following four-dimensional discrete replicator system:

$$
\begin{aligned}
x_1(t+1) &= x_1(t)\left(\frac{(\mathbf{A}\vec{x})_1}{\vec{x}^T\mathbf{A}\vec{x}}\right)\\
x_2(t+1) &= x_2(t)\left(\frac{(\mathbf{A}\vec{x})_2}{\vec{x}^T\mathbf{A}\vec{x}}\right)\\
x_3(t+1) &= x_3(t)\left(\frac{(\mathbf{A}\vec{x})_3}{\vec{x}^T\mathbf{A}\vec{x}}\right)\\
x_4(t+1) &= x_4(t)\left(\frac{(\mathbf{A}\vec{x})_4}{\vec{x}^T\mathbf{A}\vec{x}}\right)
\end{aligned}
$$

where (A) is the following fitness matrix

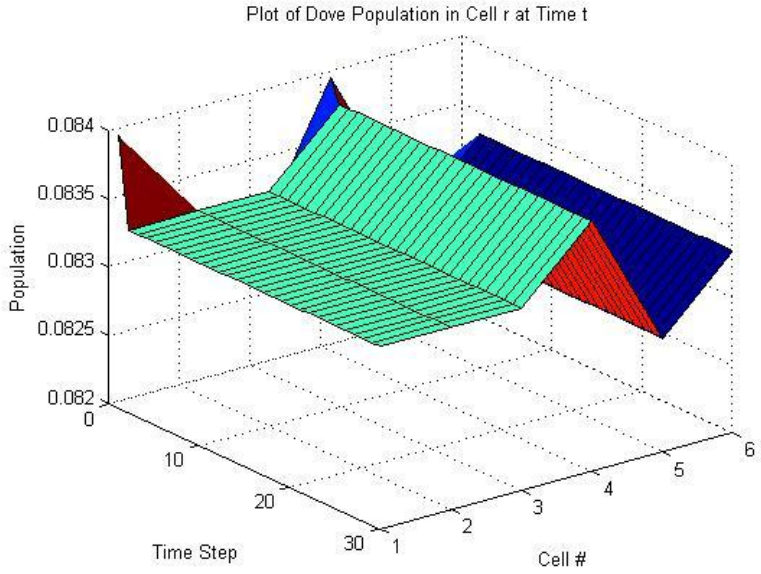
$$
\begin{bmatrix}
0 & 1+m & 0 & 1+m\\
1-m & 1 & 1 & 1-m\\
0 & 1 & 1 & 1+m\\
1-m & 1+m & 1-m & 1-m
\end{bmatrix}
$$

with $0 < m < 1$. This system describes the dynamics of interaction between four behavior tendencies: Hawks ($x_1(t)$), Doves ($x_2(t)$), Retaliators ($x_3(t)$), and Bullies ($x_4(t)$).

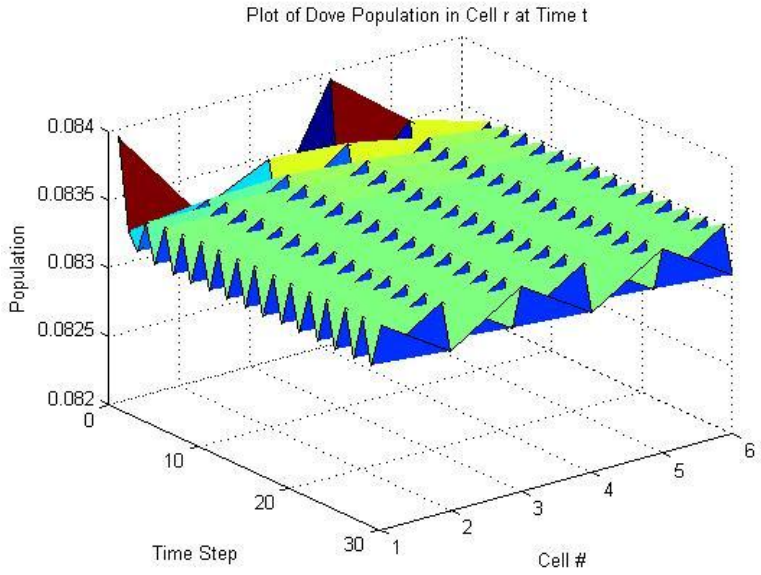
The equilibirum points of the system and their corresponding eigenvalues are given in a table below:

Equilibrium Point	Eigenvalues	Linear Stability Test
$(1, 0, 0, 0)$	Singularity (Division by 0)	Numerical results suggest instability
$(0, 1, 0, 0)$	$0,\ 1,\ 1+m,\ 1+m$	Unstable Equilbirum Point
$(0, 0, 1, 0)$	$0,\ 0,\ 1-m,\ 1$	Inconclusive by the Linear Stability Test
$(0, 0, 0, 1)$	$0,\ 1,\ \frac{-1-m}{m-1},\ \frac{-1-m}{m-1}$	Unstable Equilibrium Point
$(m, 1-m, 0)$	$0,\ \frac{1}{1+m},\ \frac{1}{1+m},\ \frac{2m+1}{1+m}$	Unstable Equilibrium Point
$(0, \alpha, 1-\alpha, 0)$ $0 < \alpha < 1$	$0,\ 1,\ \alpha(1+m),\ 1+m(2\alpha-1)$	Unstable Equilbirum Point if $\frac{1}{1+m} < \alpha < 1$
$\left(\frac{2m}{1+m}, 0, 0, \frac{1-m}{1+m}\right)$	$0,\ 1,\ 1,\ \frac{1-m}{1+m}$	Inconclusive by the Linear Stability Test

Since there is no linear temporally-stable equilibrium points in the four-dimensional case, we cannot analyze this system for instances of Turing instability.

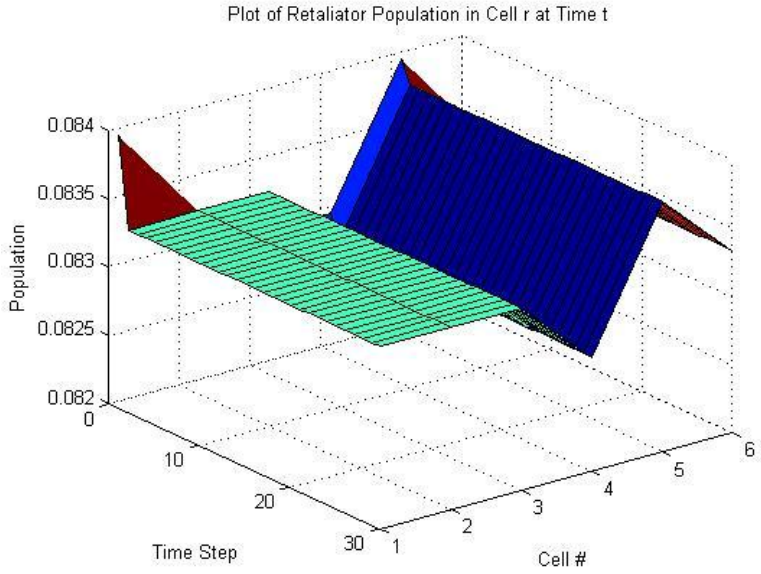


(a)

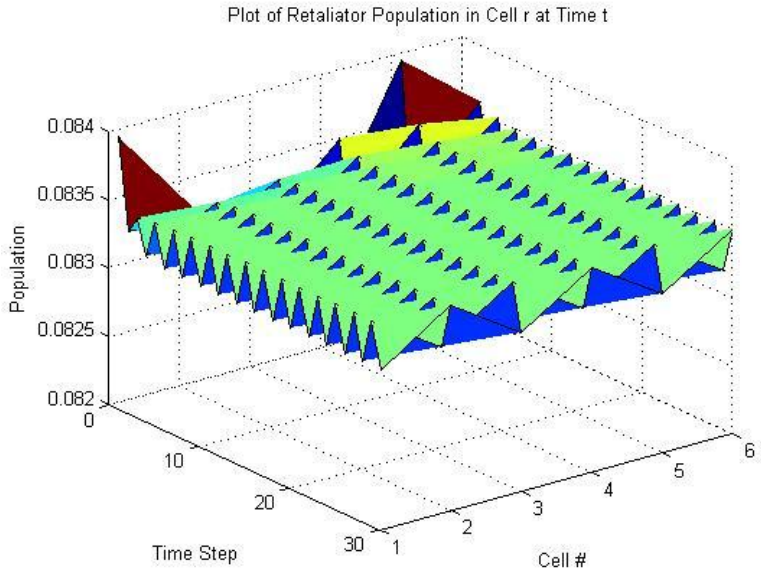


(b)

Figure 4-1: (a) A graph showing what occurs to the Dove population when diffusion is not allowed to occur in the replicator dynamic. (b) A graph showing what happens to the Dove population when diffusion is allowed to occur



(a)



(b)

Figure 4-2: (a) A graph showing what occurs to the Retaliator population when diffusion is not allowed to occur in the replicator dynamic. (b) A graph showing what happens to the Retaliator population when diffusion is allowed to occur

Chapter 5

Using Diffusion to Create Stable Equilibrium Points

Consider a related problem. Instead of looking for Turing instability, let's consider whether it is possible to use diffusion to force a temporally unstable equilibrium point to become spatially stable. Recall that the Jacobian evaluated at a temporally unstable equilibrium point has at least one eigenvalue whose magnitude is greater than 1. Thus, we want to use diffusion to force the magnitude of all eigenvalues to be less than 1. This is slightly more complex than it seems.

Recall that when we incorporate diffusion into the dynamical system, we introduced spatial cells, each of which could be considered its own dynamical system. We have to check whether diffusion will turn the eigenvalues of the unstable equilibrium point in each cell into a stable equilibrium point.

To perform this analysis, we follow a procedure similar to the approach used to check for the presence of Turing Instability in the three-dimensional case. That is, we take the Jacobian evaluated at each temporally unstable equilibrium point, incor-

porate diffusion into the matrix (by multiplying the Jacobian with the corresponding diagonal diffusion matrix), get the corresponding eigenvalues, then see what diffusion coefficients will force stability.

From the 3-d and the 4-d cases, we will be considering the following equilibrium points:

Equilibrium Point	Eigenvalues	Linear Stability Test
$(0, 1, 0)$	$0,\ 1+m,\ 1$	Unstable Equilbirum Point
$(0, \alpha, 1-\alpha)$ $0 < \alpha < 1$	$0,\ \alpha(1+m),\ 1$	Unstable Equilbirum Point if $\frac{1}{1+m} < \alpha < 1$
$(0, 1, 0, 0)$	$0,\ 1,\ 1+m,\ 1+m$	Unstable Equilbirum Point
$(0, 0, 0, 1)$	$0,\ 1,\ \frac{-1-m}{m-1},\ \frac{-1-m}{m-1}$	Unstable Equilibrium Point
$(m, 1-m, 0)$	$0,\ \frac{1}{1+m},\ \frac{1}{1+m},\ \frac{2m+1}{1+m}$	Unstable Equilibrium Point
$(0, \alpha, 1-\alpha, 0)$ $0 < \alpha < 1$	$0,\ 1,\ \alpha(1+m),\ 1+m(2\alpha-1)$	Unstable Equilbirum Point if $\frac{1}{1+m} < \alpha < 1$

For each equilibrium point, we compute (3.2) by mulitplying the Jacobian, **J** evaluated at that equilibrium point with the appropriate diagonal diffusion matrix. Next, we compute the eigenvalues of (3.2), which are listed as follows:

Equilibrium Point	Eigenvalues
$(0, 1, 0)$	$0,\ D_1(1+m),\ D_3$
$(0, \alpha, 1-\alpha)$	$0,\ D_1\alpha(1+m),\ D_3\alpha + D_2 - D_2\alpha$
$(0, 1, 0, 0)$	$0,\ D_3,\ D_1(1+m),\ D_4(1+m)$
$(0, 0, 0, 1)$	$0,\ D_2,\ \frac{-D_1(1+m)}{m-1},\ \frac{-D_3(1+m)}{m-1}$
$(m, 1-m, 0)$	$0,\ \frac{D_3}{1+m},\ \frac{D_2m+D_1-D_1m}{1+m},\ \frac{D_4(2m+1)}{1+m}$
$(0, \alpha, 1-\alpha, 0)$	$0,\ D_1\alpha(1+m),\ D_4(2\alpha+1-m),\ D_3\alpha + D_2 - D_2\alpha$

where $D_i = 1 - 4d_i \sin^2\left(\frac{r\pi}{k}\right) \in [-1, 1]$ and $d_i \in [0, 0.5]$ for $i = 1, \cdots, 4$. In order to force an temporally unstable equlibirum point to become spatially stable, the magnitude of each corresponding eigenvalue must be less than 1.

Consider the case where an equilibrium point has an eigenvalue D_i for some $i = 1, \cdots, 4$. For spatial stability, we want

$$|D_i| < 1$$

to hold for all spatial cells r, $1 \leq r \leq k$. Note that in the k^{th} cell, regardless of the d_i value, $D_i = 1$ since $\sin^2\left(\frac{k\pi}{k}\right) = 0$ It is impossible to force the unstable equilibrium point in the last (k^{th}) cell to become stable. Therefore, it is not possible for a temporally-unstable equilbirium point to become spatially stable if they have D_i as an eigenvalue. This allows us to eliminate several cases, leaving only three equilibrium points to consider.

Consider the point $(m, 1-m, 0, 0)$, which has the $\frac{D_4(2m+1)}{1+m}$ eigenvalue. For spatial stability, we want

$$\begin{aligned}
& \left|\frac{D_4(2m+1)}{1+m}\right| && < 1 \\
-1 < & \quad \frac{D_4(2m+1)}{1+m} && < 1 \\
\frac{-(1+m)}{1+2m} < & \quad D_4 && < \frac{(1+m)}{1+2m} \\
\frac{-(1+m)}{1+2m} < & \quad 1 - 4d_4 \sin^2\left(\frac{r\pi}{k}\right) && < \frac{(1+m)}{1+2m} \\
\frac{-(2+3m)}{1+2m} < & \quad -4d_4 \sin^2\left(\frac{r\pi}{k}\right) && < \frac{(-m)}{1+2m} \\
\frac{m}{4(1+2m)} < & \quad d_4 \sin^2\left(\frac{r\pi}{k}\right) && < \frac{(2+3m)}{4(1+2m)}
\end{aligned}$$

Since $0 < m < 1$, $d_4 \sin^2\left(\frac{r\pi}{k}\right)$ must be positive. However, this inequality will not

hold for the k^{th} cell, regardless of the d_4 value, since $\sin^2\left(\frac{k\pi}{k}\right) = 0$. Therefore, it is not possible to force $(m, 1-m, 0, 0)$ to become spatially stable.

Consider the two equilibirum points $(0, \alpha, 1-\alpha)$ and $(0, \alpha, 1-\alpha, 0)$, both of which have $D_1\alpha(1+m)$ as an eigenvalue. Since we are considering temporally unstable equilibrium points, $\alpha > \frac{1}{1+m}$. For spatial stability, we need

$$|D_1\alpha(1+m)| < 1$$

Since $\alpha > \frac{1}{1+m}$, let $\alpha(1+m) = x > 1$. Then we need

$$\begin{aligned}
|D_1 x| &< 1 \\
-1 < \quad D_1 x &< 1 \\
\frac{-1}{x} < \quad D_1 &< \frac{1}{x} \\
\frac{-1}{x} < \quad 1 - 4d_1 \sin^2\left(\tfrac{r\pi}{k}\right) &< \frac{1}{x} \\
\frac{-(1+x)}{x} < \quad -4d_1 \sin^2\left(\tfrac{r\pi}{k}\right) &< \frac{1-x}{x} \\
\frac{x-1}{4x} < \quad d_1 \sin^2\left(\tfrac{r\pi}{k}\right) &< \frac{1+x}{4x}
\end{aligned}$$

Since $x > 1$, $d_1 \sin^2\left(\frac{r\pi}{k}\right)$ must be positive. However, this inequality will not hold for the k^{th} cell, regardless of the d_i value, since $\sin^2\left(\frac{k\pi}{k}\right) = 0$. Hence, it is not possible to make the equilibirum points $(0, \alpha, 1-\alpha)$ and $(0, \alpha, 1-\alpha, 0)$ spatially stable using diffusion.

None of the temporally unstable equilibrium points became spatially stable. Thus, one might conclude that diffusion cannot be used to turn a temporally unstable equilibrium point a spatially stable equilibrium point.

Intuitively, this makes sense. If an equilibrium point is temporally stable, it in-

dicates that we have constant population values for all time t. Thus, it remains to see whether the spatial distribution of the population across the k cells remains constant (indicating spatial stability) or if populations continue to migrate to different cells over time, (indicating Turing instability). However, if an equilibrium point is temporally unstable, then the population values will continue to change over time. Hence, the spatial distribution of these population values will also change over time, indicating spatial instability.

Chapter 6

Conclusion

The paper aimed to discuss discrete replicator systems and to explore the parameter space for instances of Turing instability. We presented the formal definition of the discrete replicator dynamic and introduced a framework story connecting the replicator dynamic to evolutionary game theory through which we would analyze the system. The framework story enabled us to replace two parameters from the system with a dimensionless parameter. We found the equilibrium points for the 2-dimensional, 3-dimensional, and 4-dimensional replicator systems. We introduced diffusion into a general discrete dynamical system and derived a new Jacobian matrix to aid us when searching for Turing instability. Finally, we proved that Turing instability was not possible in the two-dimensional replicator system and showed it was unlikely to occur in the three-dimensional and the four-dimensional replicator systems. We also showed that it was not possible to use diffusion for a similar problem: changing a temporally unstable equilibrium point into a spatially stable equilibrium point.

Based on our work, we conjecture that it will be impossible to use diffusion to make unstable equilibrium points stable. We also conjecture that it is impossible

to cause Turing instability in the replicator system constructed in Chapter 2. Recall that when we constructed the fitness matrix, we assumed if a conflict occurs, the participants have a 50% chance of winning the fight and gaining access to the resource. It may be possible to achieve Turing instability in a different replicator system using a fitness matrix where individuals have unequal chances of winning or losing a conflict.

The possibility of unequal chances of conflict outcomes poses a new area of research. In essence, this means considering different fitness matrices and analyzing their corresponding equilibrium points. In addition, although we proved that linear Turing instability was not possible in the two-dimensional, three dimensional, and four-dimensional cases, we did provide an example of where Turing instability might occur in the three-dimensional case if we consider equilibrium points with at least one eigenvalue of magnitude 1. This presents another possible area to explore: looking beyond linear stability analysis to nonlinear stability analysis in order to find Turing instability.

Appendix A

Matlab Programs

A.1 Replicator Dynamics

A.1.1 2-dimensional

```
%Function that performs n steps of the 2d replicator system
%Inputs: parameter m, initial population [x, y], # time steps n
function[pop] = RepEqu2d(m, intpop, n)

%Initalizing the population matrix
pop = zeros(n+1, 2);
pop(1,:) = intpop;
%Constructing the fitness matrix
A = [0 (1+m); (1-m) 1];
a11 = A(1,1); a12 = A(1,2);
a21 = A(2,1); a22 = A(2,2);
```

```
for i=2:n+1
    %Constructing the denominator
    popvect = [pop(i-1,1); pop(i-1,2)];
    denom = popvect'*A*popvect;
    %Generating the population vector
    x = pop(i-1, 1)*(a11*pop(i-1,1)+a12*pop(i-1,2));
    y = pop(i-1, 2)*(a21*pop(i-1,1)+a22*pop(i-1,2));
    pop(i,1) = x/denom;
    pop(i,2) = y/denom;
    total = sum(pop(i,1)+pop(i,2));
    pop(i,:) = pop(i,:)/total;
end

%Plotting the populations
figure
plot([0:1:n], pop(:,1), [0:1:n], pop(:,2))
xlabel('Time')
ylabel('Population')
title('Population Change over Time')
legend('Hawk Population','Dove Population')
axis([0 n 0 1])

end
```

A.1.2 3-dimensional

```
%Function that performs n steps of the 3d replicator system
%Inputs: parameter m, initial population [x, y, z], # time steps n
function[pop] = RepEqu3d(m, intpop, n)

%Initalizing the population matrix
pop = zeros(n+1, 3);
pop(1,:) = intpop;
%Constructing the fitness matrix
A = [0 (1+m) 0; (1-m) 1 1; 0 1 1];
a11 = A(1,1); a12 = A(1,2); a13 = A(1,3);
a21 = A(2,1); a22 = A(2,2); a23 = A(2,3);
a31 = A(3,1); a32 = A(3,2); a33 = A(3,3);

for i=2:n+1
    %Constructing the denominator
    popvect = [pop(i-1,1); pop(i-1,2); pop(i-1,3)];
    denom = popvect'*A*popvect;
    %Generating the population vector
    x = pop(i-1, 1)*(a11*pop(i-1,1)+a12*pop(i-1,2)+a13*pop(i-1,3));
    y = pop(i-1, 2)*(a21*pop(i-1,1)+a22*pop(i-1,2)+a23*pop(i-1,3));
    z = pop(i-1, 3)*(a31*pop(i-1,1)+a32*pop(i-1,2)+a33*pop(i-1,3));
    pop(i,1) = x/denom;
    pop(i,2) = y/denom;
    pop(i,3) = z/denom;
```

```
    total = sum(pop(i,1)+pop(i,2)+pop(i,3));
    pop(i,:) = pop(i,:)/total;
end

%Plotting the populations
figure
plot([0:1:n], pop(:,1), [0:1:n], pop(:,2), [0:1:n], pop(:,3))
xlabel('Time')
ylabel('Population')
title('Population Change over Time')
legend('Hawk Population','Dove Population','Retaliator Population')
axis([0 n 0 1])

end
```

A.1.3 4-dimensional

```
%Function that performs n steps of the 4d replicator system
%Inputs: parameter m, initial population [x, y, z, w], # time steps n
function[pop] = RepEqu4d(m, intpop, n)

%Initalizing the population matrix
pop = zeros(n+1, 4);
pop(1,:) = intpop;
%Constructing the fitness matrix
A = [0, 1+m, 0, 1+m; 1-m, 1, 1, 1-m; ...
```

```
      0, 1, 1, 1+m; 1-m, 1+m, 1-m, 1-m];
a11 = A(1,1); a12 = A(1,2); a13 = A(1,3); a14 = A(1,4);
a21 = A(2,1); a22 = A(2,2); a23 = A(2,3); a24 = A(2,4);
a31 = A(3,1); a32 = A(3,2); a33 = A(3,3); a34 = A(3,4);
a41 = A(4,1); a42 = A(4,2); a43 = A(4,3); a44 = A(4,4);

for i=2:n+1
    %Constructing the denominator
    popvect = [pop(i-1,1); pop(i-1,2); pop(i-1,3); pop(i-1,4)];
    denom = popvect'*A*popvect;
    %Generating the population vector
    x = pop(i-1, 1)*(a11*pop(i-1,1)+a12*pop(i-1,2)+a13*pop(i-1,3)+...
          a14*pop(i-1,4));
    y = pop(i-1, 2)*(a21*pop(i-1,1)+a22*pop(i-1,2)+a23*pop(i-1,3)+...
          a24*pop(i-1,4));
    z = pop(i-1, 3)*(a31*pop(i-1,1)+a32*pop(i-1,2)+a33*pop(i-1,3)+...
          a34*pop(i-1,4));
    w = pop(i-1, 4)*(a41*pop(i-1,1)+a42*pop(i-1,2)+a43*pop(i-1,3)+...
           a44*pop(i-1,4));
    pop(i,1) = x/denom;
    pop(i,2) = y/denom;
    pop(i,3) = z/denom;
    pop(i,4) = w/denom;
    total = sum(pop(i,1)+pop(i,2)+pop(i,3)+pop(i,4));
    pop(i,:) = pop(i,:)/total;
end
```

```
%Plotting the popluations
figure
plot([0:1:n], pop(:,1), '-', [0:1:n], pop(:,2), ':', ...
    [0:1:n], pop(:,3), '-.', [0:1:n], pop(:,4), '--')
xlabel('Time')
ylabel('Population')
title('Population Change over Time')
legend('Hawk Population','Dove Population',...
    'Retaliator Population', 'Bully Population')
axis([0 n 0 1])

end
```

A.2 Reaction-Diffusion Replicator Dynamics

Note: in the next three programs, the inital population distribution will be entered as a m-by-n matrix where the ij^{th} entry denotes the initial value for the i^{th} behavior type in the j^{th} cell, $1 \leq j \leq n$.

A.2.1 2-dimensional

```
%Function that constructs spatial cells and
%computes population growth for 2d Replicator Dynamics
%Input: parameter m 0<m<1, initial population intpop, number of
%          interations n, diffusion coefficient vector D
```

```
function[x, y] = RepRD2d(A, intpop, n, D)

%Unpacking Diffusion coefficients
Dx = D(1); Dy = D(2);
%Error checking
if size(intpop,1) ~=2
    fprintf('Initial population vector must have only 2 rows.\n')
    return
end
if Dx<0 || Dx>.5 || Dy<0 || Dy>.5
    fprintf('Both diffusion coefficients must be within [0, .5].\n')
    return
end
%Constructing the fitness matrix
A = [0 1+m; 1-m 1];
%Gets the number of spatial cells as determined by initial pop vector
r = size(intpop, 2);
%Constructing matrices to store population values
x = zeros(n+1, r);    y = zeros(n+1, r);
%Initalizing the 0th row to represent initial distributions
x(1,:) = intpop(1,:); y(1,:) = intpop(2,:);

%Interating through n time steps
for i=2:n+1
    %Interating through r spatial cells
    for j=1:r
```

```
%Dealing with boundary conditions
if j==1
    jm1 = r;   jp1 = 2;
elseif j==r
    jm1 = r-1; jp1 = 1;
else
    jm1 = j-1; jp1 = j+1;
end
%Constructing the denominators of r, r-1, and r+1 cells
denom = [x(i-1,j) y(i-1,j)]*A*[x(i-1,j);y(i-1,j)];
denom_m1 = [x(i-1,jm1) y(i-1,jm1)]*A*[x(i-1,jm1);y(i-1,jm1)];
denom_p1 = [x(i-1,jp1) y(i-1,jp1)]*A*[x(i-1,jp1);y(i-1,jp1)];
%Computing the population values
x(i,j) = x(i-1,j)*(A(1,1)*x(i-1,j) + A(1,2)*y(i-1,j));
x(i,j) = x(i,j)/denom;
y(i,j) = y(i-1,j)*(A(2,1)*x(i-1,j) + A(2,2)*y(i-1,j));
y(i,j) = y(i,j)/denom;
%Computing diffusion terms for the x population
if Dx ~= 0
    %Computing the r-1 cell population
    xm1 = x(i-1,jm1)*(A(1,1)*x(i-1,jm1) + A(1,2)*y(i-1,jm1));
    xm1 = xm1/denom_m1;
    %Computing the r+1 cell populations
    xp1 = x(i-1,jp1)*(A(1,1)*x(i-1,jp1) + A(1,2)*y(i-1,jp1));
    xp1 = xp1/denom_p1;
    %Computing Diffusion terms
```

```
            xdiff = Dx*(xm1 - 2*x(i,j) + xp1);
            %Adding Diffusion terms to r cell populations
            x(i,j) = x(i,j) + xdiff;
        end
        %Computing diffusion terms for the y population
        if Dy ~= 0
            %Computing the r-1 cell population
            ym1 = y(i-1,jm1)*(A(2,1)*x(i-1,jm1) + A(2,2)*y(i-1,jm1));
            ym1 = ym1/denom_m1;
            %Computing the r+1 cell population
            yp1 = y(i-1,jp1)*(A(2,1)*x(i-1,jp1) + A(2,2)*y(i-1,jp1));
            yp1 = yp1/denom_p1;
            %Computing Diffusion terms
            ydiff = Dy*(ym1 - 2*y(i,j) + yp1);
            %Adding Diffusion terms to r cell populations
            y(i,j) = y(i,j) + ydiff;
        end
    end %End of loop through space
    %Rescale the population vector so it adds up to 1
    total = sum(x(i,:)) + sum(y(i,:));
    x(i,:) = x(i,:)/total;
    y(i,:) = y(i,:)/total;
end %End of loop through time

%Generating 3d plots to show population values
figure
```

```
subplot(1, 2, 1)
surf(x([n+1:-1:1],:))
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Population X in Cell r at Time t')
subplot(1,2,2)
surf(y([n+1:-1:1],:))
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Population Y in Cell r at Time t')
```

A.2.2 3-dimensional

```
%Function for constructing spatial cells and
%computing population growth for 3d replicator dynamics
%Input: parameter m, 0<m<1, initial population intpop,
%         # time steps n, diffusion coefficient vector [d_1, d_2, d_3]
function[x, y, z] = RepRD3d(m, intpop, n, D)

%Unpacking Diffusion coefficients
Dx = D(1); Dy = D(2); Dz = D(3);
%Error checking
if m<0 || m>1
```

```
    fprintf('m must be between 0 and 1 exclusive.\n')
    return
end
if size(intpop,1) ~=3
    fprintf('Initial population vector must have only 3 rows.\n')
    return
end
if Dx<0 || Dx>.5 || Dy<0 || Dy>.5 || Dz<0 || Dz>.5
    fprintf('All diffusion coefficients must be within [0, .5].\n')
    return
end
%Constructing the fitness matrix
A = [0 1+m 0; 1-m 1 1; 0 1 1];
%Gets the number of spatial cells determined by initial pop vector
r = size(intpop, 2);
%Constructing matrices to store population values
x = zeros(n+1, r);    y = zeros(n+1, r);    z = zeros(n+1, r);
%Initalizing the 0th row to represent initial distributions
x(1,:) = intpop(1,:); y(1,:) = intpop(2,:); z(1,:) = intpop(3,:);

%Interating through n time steps
for i=2:n+1
    %Interating through r spatial cells
    for j=1:r
        %Dealing with boundary conditions
        if j==1
```

```
    jm1 = r;   jp1 = 2;
 elseif j==r
    jm1 = r-1; jp1 = 1;
 else
    jm1 = j-1; jp1 = j+1;
 end
%Constructing the denominators of r, r-1, and r+1 cells
 denom = [x(i-1,j),y(i-1,j),z(i-1,j)]*A*...
              [x(i-1,j);y(i-1,j);z(i-1,j)];
 denom_m1 = [x(i-1,jm1),y(i-1,jm1),z(i-1,jm1)]*A*...
                   [x(i-1,jm1);y(i-1,jm1);z(i-1,jm1)];
 denom_p1 = [x(i-1,jp1),y(i-1,jp1),z(i-1,jp1)]*A*...
                  [x(i-1,jp1);y(i-1,jp1);z(i-1,jp1)];
 %Computing the population values
 x(i,j) = x(i-1,j)*(A(1,1)*x(i-1,j) + A(1,2)*y(i-1,j) ...
           + A(1,3)*z(i-1,j));
 x(i,j) = x(i,j)/denom;
 y(i,j) = y(i-1,j)*(A(2,1)*x(i-1,j) + A(2,2)*y(i-1,j) ...
           + A(2,3)*z(i-1,j));
 y(i,j) = y(i,j)/denom;
 z(i,j) = z(i-1,j)*(A(3,1)*x(i-1,j) + A(3,2)*y(i-1,j) ...
           + A(3,3)*z(i-1,j));
 z(i,j) = z(i,j)/denom;
 %Computing diffusion terms for the x population
 if Dx ~= 0
    %Computing the r-1 cell population
```

```
    xm1 = x(i-1,jm1)*(A(1,1)*x(i-1,jm1) + ...
              A(1,2)*y(i-1,jm1) + A(1,3)*z(i-1,jm1));
    xm1 = xm1/denom_m1;
    %Computing the r+1 cell populations
    xp1 = x(i-1,jp1)*(A(1,1)*x(i-1,jp1) + ...
             A(1,2)*y(i-1,jp1) + A(1,3)*z(i-1,jp1));
    xp1 = xp1/denom_p1;
    %Computing Diffusion terms
    xdiff = Dx*(xm1 - 2*x(i,j) + xp1);
    %Adding Diffusion terms to r cell populations
    x(i,j) = x(i,j) + xdiff;
end
%Computing diffusion terms for the y population
if Dy ~= 0
    %Computing the r-1 cell population
    ym1 = y(i-1,jm1)*(A(2,1)*x(i-1,jm1) + ...
              A(2,2)*y(i-1,jm1) + A(2,3)*z(i-1,jm1));
    ym1 = ym1/denom_m1;
    %Computing the r+1 cell population
    yp1 = y(i-1,jp1)*(A(2,1)*x(i-1,jp1) + ...
             A(2,2)*y(i-1,jp1) + A(2,3)*z(i-1,jp1));
    yp1 = yp1/denom_p1;
    %Computing Diffusion terms
    ydiff = Dy*(ym1 - 2*y(i,j) + yp1);
    %Adding Diffusion terms to r cell populations
    y(i,j) = y(i,j) + ydiff;
```

```
        end
        %Computing diffusion term for the z population
        if Dz ~= 0
            %Computing the r-1 cell population
            zm1 = z(i-1,jm1)*(A(3,1)*x(i-1,jm1) + ...
                      A(3,2)*y(i-1,jm1) + A(3,3)*z(i-1,jm1));
            zm1 = zm1/denom_m1;
            %Computing the r+1 cell population
            zp1 = z(i-1,jp1)*(A(3,1)*x(i-1,jp1) + ...
                     A(3,2)*y(i-1,jp1) + A(3,3)*z(i-1,jp1));
            zp1 = zp1/denom_p1;
            %Computing Diffusion terms
            zdiff = Dz*(zm1 - 2*z(i,j) + zp1);
            %Adding Diffusion terms to r cell populations
            z(i,j) = z(i,j) + zdiff;
        end
    end %End of loop through space
    %Rescale the population vector so it adds up to 1
    total = sum(x(i,:)) + sum(y(i,:)) + sum(z(i,:));
    x(i,:) = x(i,:)/total;
    y(i,:) = y(i,:)/total;
    z(i,:) = z(i,:)/total;
end %End of loop through time

%Plots popluation growth for each type over all cells
figure
```

```
surf(x([1:1:n],:))
set(gca,'YDir','reverse')
%AXIS([1 r 0 n 0 .15])
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Hawk Population in Cell r at Time t')
hold on
figure
surf(y([1:1:n],:))
set(gca,'YDir','reverse')
%AXIS([1 r 0 n .1 .15])
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Dove Population in Cell r at Time t')
hold on
figure
surf(z([1:1:n],:))
set(gca,'YDir','reverse')
%AXIS([1 r 0 n .1 .15])
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Retaliator Population in Cell r at Time t')
hold on
```

```
%Plots population dynamics in each cell
for i=1:r
    figure
    plot([0:1:n], x(:, i), [0:1:n], y(:,i), [0:1:n], z(:,i))
    xlabel('Time')
    ylabel('Population')
    title('Popluation in the cell')
    legend('Hawk Population','Dove Population',...
               'Retaliator Population')
    axis([0 n 0 1])
    hold on
end
```

A.2.3 4-dimensional

```
%Function for constructing spatial cells and
%computing the population growth for 4d repliator dynamics
%Input: parameter m, 0<m<1, initial population intpop,
%         # time steps n, diffusion coefficient [d_1, d_2, d_3, d_4]
function[x, y, z, w] = RepRD4d(m, intpop, n, D)

%Unpacking Diffusion coefficients
Dx = D(1); Dy = D(2); Dz = D(3); Dw = D(4);
%Error checking
if m<0 || m>1
    fprintf('m must be between 0 and 1 exclusive.\n')
```

```
    return
end
if size(intpop,1) ~=4
    fprintf('Initial population vector must have only 4 rows.\n')
    return
end
if Dx<0 || Dx>.5 || Dy<0 || Dy>.5 || Dz<0 || Dz>.5 || Dw<0 || Dw>.5
    fprintf('All diffusion coefficients must be within [0, .5].\n')
    return
end
%Constructing the matrix)
A = [0 1+m 0 1+m; 1-m 1 1 1-m; 0 1 1 1+m; 1-m 1+m 1-m 1-m];
%Gets the number of spatial cells determined by initial pop vector
r = size(intpop, 2);
%Constructing matrices to store population values
x = zeros(n+1, r);    y = zeros(n+1, r);
z = zeros(n+1, r);    w = zeros(n+1, r);
%Initalizing the 0th row to represent initial distributions
x(1,:) = intpop(1,:); y(1,:) = intpop(2,:);
z(1,:) = intpop(3,:); w(1,:) = intpop(4,:);

%Interating through n time steps
for i=2:n+1
    %Interating through r spatial cells
    for j=1:r
        %Dealing with boundary conditions
```

```
 if j==1
     jm1 = r;   jp1 = 2;
 elseif j==r
     jm1 = r-1; jp1 = 1;
 else
     jm1 = j-1; jp1 = j+1;
 end
%Constructing the denominators of r, r-1, and r+1 cells
 denom = [x(i-1,j),y(i-1,j),z(i-1,j),w(i-1,j)]*A*...
             [x(i-1,j);y(i-1,j);z(i-1,j);w(i-1,j)];
 denom_m1 =[x(i-1,jm1),y(i-1,jm1),z(i-1,jm1),w(i-1,jm1)]...
             *A*[x(i-1,jm1);y(i-1,jm1);z(i-1,jm1);w(i-1,jm1)];
 denom_p1 = [x(i-1,jp1),y(i-1,jp1),z(i-1,jp1),w(i-1,jp1)]...
            *A*[x(i-1,jp1); y(i-1,jp1); z(i-1,jp1);w(i-1,jp1)];
 %Computing the population values
 x(i,j) = x(i-1,j)*(A(1,1)*x(i-1,j) + A(1,2)*y(i-1,j) ...
           + A(1,3)*z(i-1,j) + A(1,4)*w(i-1,j));
 x(i,j) = x(i,j)/denom;
 y(i,j) = y(i-1,j)*(A(2,1)*x(i-1,j) + A(2,2)*y(i-1,j) ...
           + A(2,3)*z(i-1,j) + A(2,4)*w(i-1,j));
 y(i,j) = y(i,j)/denom;
 z(i,j) = z(i-1,j)*(A(3,1)*x(i-1,j) + A(3,2)*y(i-1,j) ...
           + A(3,3)*z(i-1,j) + A(3,4)*w(i-1,j));
 z(i,j) = z(i,j)/denom;
 w(i,j) = w(i-1,j)*(A(4,1)*x(i-1,j) + A(4,2)*y(i-1,j) ...
             + A(4,3)*z(i-1,j) + A(4,4)*w(i-1,j));
```

```
w(i,j) = w(i,j)/denom;
%Computing diffusion terms for the x population
if Dx ~= 0
    %Computing the r-1 cell population
    xm1 = x(i-1,jm1)*(A(1,1)*x(i-1,jm1) + ...
          A(1,2)*y(i-1,jm1) + A(1,3)*z(i-1,jm1) + ...
          A(1,4)*w(i-1,jm1));
    xm1 = xm1/denom_m1;
    %Computing the r+1 cell populations
    xp1 = x(i-1,jp1)*(A(1,1)*x(i-1,jp1) + ...
          A(1,2)*y(i-1,jp1) + A(1,3)*z(i-1,jp1) + ...
          A(1,4)*w(i-1,jp1));
    xp1 = xp1/denom_p1;
    %Computing Diffusion terms
    xdiff = Dx*(xm1 - 2*x(i,j) + xp1);
    %Adding Diffusion terms to r cell populations
    x(i,j) = x(i,j) + xdiff;
end
%Computing diffusion terms for the y population
if Dy ~= 0
    %Computing the r-1 cell population
    ym1 = y(i-1,jm1)*(A(2,1)*x(i-1,jm1) + ...
          A(2,2)*y(i-1,jm1) + A(2,3)*z(i-1,jm1) + ...
          A(2,4)*w(i-1,jm1));
    ym1 = ym1/denom_m1;
    %Computing the r+1 cell population
```

```
    yp1 = y(i-1,jp1)*(A(2,1)*x(i-1,jp1) + ...
          A(2,2)*y(i-1,jp1) + A(2,3)*z(i-1,jp1) +...
          A(2,4)*w(i-1,jp1));
    yp1 = yp1/denom_p1;
    %Computing Diffusion terms
    ydiff = Dy*(ym1 - 2*y(i,j) + yp1);
    %Adding Diffusion terms to r cell populations
    y(i,j) = y(i,j) + ydiff;
end
%Computing diffusion term for the z population
if Dz ~= 0
    %Computing the r-1 cell population
    zm1 = z(i-1,jm1)*(A(3,1)*x(i-1,jm1) + ...
          A(3,2)*y(i-1,jm1) + A(3,3)*z(i-1,jm1) +...
          A(3,4)*w(i-1,jm1));
    zm1 = zm1/denom_m1;
    %Computing the r+1 cell population
    zp1 = z(i-1,jp1)*(A(3,1)*x(i-1,jp1) + ...
          A(3,2)*y(i-1,jp1) + A(3,3)*z(i-1,jp1) + ...
          A(3,4)*w(i-1,jp1));
    zp1 = zp1/denom_p1;
    %Computing Diffusion terms
    zdiff = Dz*(zm1 - 2*z(i,j) + zp1);
    %Adding Diffusion terms to r cell populations
    z(i,j) = z(i,j) + zdiff;
end
```

```
        %Computing diffusion term for the w population
        if Dw ~= 0
            %Computing the r-1 cell population
            wm1 = w(i-1,jm1)*(A(4,1)*x(i-1,jm1) + ...
                  A(4,2)*y(i-1,jm1) + A(4,3)*z(i-1,jm1) + ...
                  A(4,4)*w(i-1,jm1));
            wm1 = wm1/denom_m1;
            %Computing the r+1 cell population
            wp1 = w(i-1,jp1)*(A(4,1)*x(i-1,jp1) + ...
                  A(4,2)*y(i-1,jp1) + A(4,3)*z(i-1,jp1) + ...
                  A(4,4)*w(i-1,jp1));
            wp1 = wp1/denom_p1;
            %Computing Diffusion terms
            wdiff = Dw*(wm1 - 2*w(i,j) + wp1);
            %Adding Diffusion terms to r cell populations
            w(i,j) = w(i,j) + wdiff;
        end
    end %End of loop through space
    %Rescale the population vector so it adds up to 1
    total = sum(x(i,:)) + sum(y(i,:)) + sum(z(i,:)) + sum(w(i,:));
    x(i,:) = x(i,:)/total;
    y(i,:) = y(i,:)/total;
    z(i,:) = z(i,:)/total;
    w(i,:) = w(i,:)/total;

end %End of loop through time
```

```
%Plots population growth over all cells
figure
surf(x([1:1:n+1],:))
set(gca,'YDir','reverse')
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Hawk Population in Cell r at Time t')
hold on
figure
surf(y([1:1:n+1],:))
set(gca,'YDir','reverse')
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Dove Population in Cell r at Time t')
hold on
figure
surf(z([1:1:n+1],:))
set(gca,'YDir','reverse')
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Retaliator Population in Cell r at Time t')
hold on
```

```
figure
surf(w([1:1:n+1],:))
set(gca,'YDir','reverse')
ylabel('Time Step')
xlabel('Cell #')
zlabel('Population')
title('Plot of Bully Population in Cell r at Time t')
hold on
%Plots population dynamics for each cell
for i=1:r
    figure
    plot([0:1:n], x(:, i), [0:1:n], y(:,i), [0:1:n], z(:,i), ...
    [0:1:n], w(:,i))
    xlabel('Time')
    ylabel('Population')
    title('Popluation in the cell')
    legend('Hawk Population','Dove Population',...
              'Retaliator Population', 'Bully Population')
    axis([0 n 0 1])
    hold on
end
```

Bibliography

[1] Immanuel M. Bomze, *Lotka-volterra equation and replicator dynamics: new issues in classification*, Biological Cybernetics **72** (1995), 447–453.

[2] Bernard P. Brooks, *Turing instabilities in a diploid population with frequency and density dependent selection*, 2000.

[3] ———, *Linear stability conditions for a first order 3-dimensional discrete dynamic*, 2002.

[4] Josef Hofbauer and Karl Sigmund, *The theory of evolution and dynamical systems*, Cambridge University Press, 1988.

[5] ———, *Evolutionary game dynamics*, Bulletin of the American Mathematical Society **40.4** (2003), 479–519.

[6] L. Perko, *Differential equations and dynamical systems*, Springer-Verlag, New York, 1991.

[7] A.M. Turing, *The chemical basis of morphogenesis*, Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences **237.641** (1952), 37–72.

[8] Jorgen W. Weibull, *Evolutionary game theory*, MIT Press, Cambridge, Massachusetts, 1998.