BREEDING POND DISPERSAL OF INTERACTING CALIFORNIA RED-LEGGED FROGS (*RANA DRAYTONII*) AND AMERICAN BULLFROGS (*LITHOBATES CATESBEIANUS*) OF CALIFORNIA: A MATHEMATICAL MODEL WITH MANAGEMENT STRATEGIES

HUMBOLDT STATE UNIVERSITY

By

Iris Acacia Gray

A Thesis

Presented to

The Faculty of Humboldt State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

In Environmental Systems: Mathematical Modeling

December, 2009

BREEDING POND DISPERSAL OF INTERACTING CALIFORNIA RED-LEGGED FROGS (*RANA DRAYTONII*) AND AMERICAN BULLFROGS (*LITHOBATES CATESBEIANUS*) OF CALIFORNIA: A MATHEMATICAL MODEL WITH MANAGEMENT STRATEGIES

HUMBOLDT STATE UNIVERSITY

By

Iris Acacia Gray

Approved by the Master's Thesis Committee:

Dr. Borbala Mazzag, Major Professor	Date
Dr. Christopher Dugaw, Committee Member	Date
Dr. Sharyn Marks, Committee Member	Date
Dr. Christopher Dugaw, Graduate Coordinator	Date
Jená Burgess, Vice Provost Research, Graduate Studies & International Programs	Date

ABSTRACT

BREEDING POND DISPERSAL OF INTERACTING CALIFORNIA RED-LEGGED FROGS (*RANA DRAYTONII*) AND AMERICAN BULLFROGS (*LITHOBATES CATESBEIANUS*) OF CALIFORNIA: A MATHEMATICAL MODEL WITH MANAGEMENT STRATEGIES

Iris Acacia Gray

The invasion of American Bullfrogs (*Lithobates catesbeianus*) in the western United States has had a direct negative effect on the persistence of many native fauna. Native ranid frogs, in particular, face threats to their populations due to the competition and predation imposed by this behaviorally similar invader. The California Red-legged Frog (*Rana draytonii*) has been listed as a threatened species since 1996 due in part to the introduction of American Bullfrogs. A stage-based modeling effort has been developed for interacting American Bullfrogs and California Red-legged Frogs within a single pond (Doubledee et al., 2003). We modified and expanded this model to encompass several ponds by implementing juvenile dispersal between them. Movement rules were created using assumptions based in part on dispersal data collected for a California Red-legged Frog population within eight specific ponds in the Marin County area. They incorporate the probability of the juvenile population being able to disperse a given distance and the probability of choosing to move to a specific pond when several possibilities are present. We performed parameter studies for three unknown parameters: predation of tadpole and juvenile California Red-legged Frogs by adult American Bullfrogs and predation/competition of overwintered Bullfrog tadpoles on/with California Red-legged Frog eggs. This study provides combinations of these rates which facilitate coexistence between these species over a specified duration within the study ponds. Given a specific duration of coexistence (60 years) we

explored the effect of bullfrog tadpole, metamorph, and juvenile/adult focused eradication efforts as management strategies. The optimal rates of eradication associated with each of these tactics benefiting California Red-legged Frog persistence were established. We determined that seasonal ponds need not be managed, as they act as a refuge to the California Red-legged frog, and thus aid in its long term survival. We found that removing at least 75% of the American Bullfrog tadpole population every year or draining ponds (100% Bullfrog tadpole eradication) at least every two years are sufficient strategies for managing all permanent ponds to achieve population levels exhibited by seasonal ponds. However, our model showed that without management permanent ponds connected (within a predetermined dispersal range) to seasonal ponds were able to harbor some population of California Red-legged frog beyond our assumed value of 60 years. Furthermore, we outline the areas in which further research could be implemented to enhance our model.

Key words: California Red-legged Frog, American Bullfrog, stage-based model, dispersal, coexistence, management.

ACKNOWLEDGMENTS

First and foremost, I would like to thank the members of my thesis committee. Dr. Bori Mazzag, Dr. Chris Dugaw, and Dr. Sharyn Marks were invaluable fixtures within this project. Special thanks to Dr. Sharyn Marks for sharing her knowledge of amphibians and for her careful and thorough edits. Through the few meetings we had discussing the biology and ecology of these frogs, she enabled me to gain an understanding of their world I perhaps could not have gathered on my own. Special thanks to Dr. Chris Dugaw for requiring me to learn the computer programming language LaTeX. This skill will aid me in achieving my research goals for the rest of my life, throughout which I'll always be thankful to him. He was also quite helpful in removing barriers in understanding some the more sophisticated mathematical and statistical elements involved in completing this project. Among my committee members, I am most thankful to the Chair of my committee: Dr. Bori Mazzag. Special thanks goes out to her for her editing, her help in the project's basic design development and understanding of the mathematics involved, and her outstanding knowledge of the programming language Matlab throughout the entire course of this project. Furthermore, Bori was much more to me than just an advisor, she also occasionally played the role of counselor. When personal problems plagued me, she was always there to listen, offer advice, kleenex, and big warm hug. For all that she's done for me, I fear I'll never be able to thank her properly.

The experiences I've shared with my colleagues over the years spent working on my Master's Degree and this project are immeasurably precious to me. I would like to especially thank Emily Hobelmann, Ben Holt, Chris Panza, Liz Arnold, Nicole Batenhorst, Thé Thé Kyaw, Daniele Rosa, Holly Perryman, Michelle Moreno, Kyle Falbo, Michael Omstead, and Rick Warnock in this respect. They've laughed with me, they've struggled with

me, and most importantly they've done mathematics with me. Because of this, these people will always be in my heart.

There were several professors which gave me support and guidance beyond the scope of their post. I would like to thank Dr. Sharon Brown, Professor Tim Lauck, Dr. Steve McKelvey, Professor Steve Wright, and Dr. Charles Biles. These individuals would always give me opportunities to talk about my project and provided help whenever possible.

When I started getting serious about the topic of my thesis project, Dr. Sharyn Marks suggested I join the HSU club Herpetology Group. I am so thankful to Sharyn for this suggestion and to the individuals involved for providing me with so much help in understanding the biology, behavior, and ecology involved with these particular species. I would like to thank Dr. Hartwell Welsh, Jamie Bettaso, Dr. W. Bryan Jennings, Oliver Miano, Ryan Bourque, Justin Garwood, Karen Pope, Jada Howarth, Mike Best, Elijah Best, Shannon Chapin, Luke Groff, Mike van Hattem, Cheryl Bondi, and Jen Brown.

I would also like to thank my awesome family for their unwavering interest and support of my academic pursuits. My mom Janice Gray, my dad Paul Gray, my sisters Jennifer Martinez and Erica Avila, my brother-in-laws Anthony Martinez and Jered Avila, my nephew Jacob, and my niece Allorah have been so incredibly understanding of my absence during this long processes. Thanks also goes out to my good friends back home Erica, CJ, and now little Charlotte Lassen for their encouragement and support.

For employing me over the entire course of my Master's work, I would like to thank Joe Doherty, current owner, and Fukiko Marshall, previous owner of Tomo Japanese Restaurant Inc. Joe has been such an fantastic boss and friend. He even let me leave Tomo for an entire semester in order to focus more on my thesis. My coworkers at this establishment are not merely coworkers to me, they are my Tomo family. I would like to thank in particular Ellie Woods, Rachel Baker de Kater, Samuel 'Man Sam' Hamilton, Jeremiah Hammond,

Nathan Rushton, Ryan Roberts, Vaden Hoffman, Aislinn Anderson, Bryce Peebles, and Rex Garland for allowing me to yammer on and on about my thesis topic while we worked together.

Against his wishes, I would like to give individual thanks to Oliver Miano. Long before I formulated the topic of this thesis project, Oliver was the first person to inform me of the bullfrog problem. He had just acquired a male albino bullfrog. We were admiring it in the backyard of our house on Zelia Court as he told me of how the introduction of bullfrogs have become a problem on the West Coast of the U.S. through competition with and predation of native frogs. Little did I realize that years later I would have the opportunity to model this very problem and achieve a Master's degree in the process. I have learned so much from Oliver about herpetology and so many other topics. I am incredibly lucky to have, and very thankful for, his presence in my life. I therefore dedicate this body of work to him.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The decline of native frog populations has become a significant ecological problem throughout western North America (Chatwin and Govindarajulu, 2006; Hayes and Jennings, 1986; Alford and Richards, 1999). Explanations for their decline are plentiful and include, but are not limited to, the effects of temperature, UV radiation, pesticides, habitat destruction and/or fragmentation, competition, predation, and invasion of exotic species (Blaustein and Bancroft, 2007; Blaustein et al., 2003; Franklin et al., 2002; Kats and Ferrer, 2003; Davidson et al., 2001; Alford and Richards, 1999; Blaustein and Kiesecker, 1998; Hayes and Jennings, 1986). Evident in all of the studies that have explored these impacts is the need for further data collection, research, and the expansion of current models.

In our work, we have chosen to model the effects of the invasion of an exotic species (*Lithobates catesbeianus*, American Bullfrog) on a native species (*Rana draytonii*, California Red-legged Frog) and its contribution to the decline of the native species, which often results in serious ecological consequences (Alford and Richards, 1999; Hayes and Jennings, 1986; Adams, 1999; Moyle, 1973). The California Red-legged Frog is of particular interest to conservationists, as it has been classified as a threatened species since 1993 through the Endangered Species Act (Doubledee et al., 2003; Blaustein and Kiesecker, 1998; Kiesecker and Blaustein, 1997; Zonick, 2005; Davidson et al., 2001; Moyle, 1973). Although presence of predatory fish may be a greater threat to the persistence of this native species, Bullfrog presence is also considered to have a significant impact (Lannoo, 2005). Thus, obtaining a clearer understanding of the effects of bullfrog presence will aide in determining what is needed to stabilize California Red-legged Frog populations.

The California Red-legged Frog ranges from southern California, along the coastline, bounded by the Sierra Nevada mountain range, up to the southern region of northern Cal-

ifornia; while the American Bullfrog is native to eastern North America (Lannoo, 2005). California Red-legged Frogs and American Bullfrogs are ranid frogs, or *true frogs*, which means that both species are similar in anatomy and physiology (Doubledee et al., 2003; Zonick, 2005; Hayes and Jennings, 1986). They exhibit a complex life cycle with multiple stages: egg, tadpole, metamorph, juvenile, and adult (Lannoo, 2005). Both species use aquatic and terrestrial habitats throughout the course of their development (Lannoo, 2005). Clearly, these species will have to compete with each other in the event they share a given area.

Bullfrogs were first transported to the West Coast of North America from their native East Coast around the time of the Gold Rush (mid 1800s) to satisfy the appetite of French cuisine enthusiasts craving frog legs. Before the introduction of bullfrogs, Californians were harvesting Red-legged Frogs from the wild in order to keep up with demand, until they were all but completely eliminated (Jennings and Hayes, 1985; Hayes and Jennings, 1986). Since their introduction, bullfrogs have impacted the persistence of many native species of frogs and other fauna by altering the structure of the ecosystems in which they inhabit (Blaustein and Kiesecker, 1998). For instance, when an invasive frog species eliminates a native frog species, the natural predator for that native frog may try to prey upon the invaders if the exotic has managed to take over the niche carved out by the native frog. Whether or not the new inhabitant of that niche is able to become part of the diet of that natural predator will determine the intensity to which the food web will be affected. Furthermore, the eating habits and degree of proliferation of the invasive frog will determine which lower trophic level species' populations linger, die out, or grow due to the lack of control from the absent native frog predator (Herbold and Moyle, 1986).

The American Bullfrog has a significant size advantage over the California Red-legged Frog (Doubledee et al., 2003). The Red-legged Frog averages 2 to 5.25 inches in length

while the American Bullfrog averages 4 to 8 inches (Lannoo, 2005). This greater size allows the Bullfrog to produce more offspring and travel further distances than the Red-legged Frog (Lannoo, 2005). Furthermore, Bullfrogs are opportunistic predators, devouring anything that can fit into their large mouths (Govindarajulu et al., 2005). The average adult Bullfrog could devour even the largest adult Red-legged Frog with little difficulty.

Eggs, tadpoles, and metamorphs of the California Red-legged Frog also face hardship as a result of Bullfrog presence (Kiesecker and Blaustein, 1997). When these two species share a pond, Bullfrog tadpoles are presented with little competition for nutrients, as Red-legged Frog tadpoles are smaller in size and number (Kupferberg, 1997; Kiesecker and Blaustein, 1997; Doubledee et al., 2003). This enables Bullfrog tadpoles to monopolize resources within the pond, regardless of the fact that California Red-legged Frog clutches are laid earlier in the year. Field experiments demonstrate that once Red-legged Frog eggs are oviposited in their natal pond, they run the risk of being nibbled by the large overwintered (survived in a permanent pond through the winter season) Bullfrog tadpoles (Kupferberg, 1997).

The only modeling effort to address the problem of California Red-legged Frog persistence despite Bullfrog presence was put forth by Doubledee et al. (2003). The authors developed a stage-based model that describes the complex life cycles of both species and their interactions in a single pond. Doubledee et al. (2003) implemented simulations using two coupled matrix equations to illuminate the effect of Bullfrog eradication efforts such as tadpole removal via pond draining and shooting of adult frogs. Doubledee et al. (2003) conclude that shooting adults alone would require an exorbitant amount of effort, and is therefore not a feasible management strategy. Govindarajulu et al. (2005) performed an empirical study on invasive bullfrog population dynamics on Vancouver Island and found that culling bullfrogs at the metamorphic stage had the greatest impact in reducing their

total population growth rate.

Doubledee et al. (2003) concluded that the next logical step in the modeling process would be to add a spatial component allowing movement between ponds with information about habitat condition and complexity. In our extension of the Doubledee et al. (2003) model, we are concerned primarily with space. Space is represented by discrete patches which will represent the ponds that the frogs would use for breeding. We used telemetry data collected by Fellers and Kleeman (2007) for the California Red-legged Frog over eight ponds within Point Reyes National Seashore in Marin County, CA. We created movement rules which allow the frogs of both species to disperse between these ponds based on the pond's proximity to other ponds, seasonality, population size, and also dependent on the life history stage of the frogs themselves.

Since complete eradication of the American Bullfrog from non-native areas at this juncture in its invasion would be monumentally lengthy, costly and realistically infeasible, it would be in the best interest of wildlife managers to implement strategies for coexistence (Manchester and Bullock, 2000; Hayes and Jennings, 1986). In the analysis of our new model, we determine which of three management strategies to control Bullfrogs (tadpole, metamorph, and juvenile/adult focused eradictions) is most effective in terms of facilitating coexistence between our study species in a multiple pond habitat.

METHODS

One Pond Model

The foundation of our model is taken from a dual stage-based model presented by Doubledee et al. (2003). This model follows the life cycles of both the American Bullfrog and the California Red-legged Frog as they co-habitate a single permanent pond. The number of individuals in eahc stage as a function of time is described by an equation. Over both systems, each equation is discrete, updating every year of the simulation. Since both species breed once a year, a discrete system is most appropriate for modeling both of these species. For the sake of brevity, the California Red-legged Frog will be referred to as CRLF and the American Bullfrog will be denoted as bullfrog.

We updated the Doubledee et al. (2003) model with the intention of increasing its accuracy. Their bullfrog system (composed of three equations, or stages) is smaller than the CRLF system (composed of four equations, or stages). According to their model the bullfrogs in the simulation reach adulthood earlier than CRLFs. Normally, bullfrogs take four to five years to reach sexual maturity while the CRLF takes approximately two to three years (Lannoo, 2005). Thus, we added two more equations to the bullfrog system, making it a five stage system. This expansion of the bullfrog system also allows for a 'fast track' option to be incorporated for the first year tadpole stage so that individual tadpoles can enter metamorphose in only one year (rather than two) if conditions in the pond are optimal for larval development (Govindarajulu et al., 2005).

Based on the biology of the CRLF, we collapsed the two juvenile stages outlined in the Doubledee et al. (2003) model into a single stage. When CRLF clutches are laid in April, the only other stages that are present in the habitat are the second year juveniles and adults; the first year juveniles do not reveal themselves until later that same year (about

September to October). This led us to absorb the first year juvenile stage (as it is known in the Doubledee et al. (2003) model) into the tadpole stage of the model (see our timeline in Figure 1). Furthermore, since the census occurs during the birth pulse, we are only concerned with modeling female frogs. This maintains the convention established by the original model (Doubledee et al., 2003).

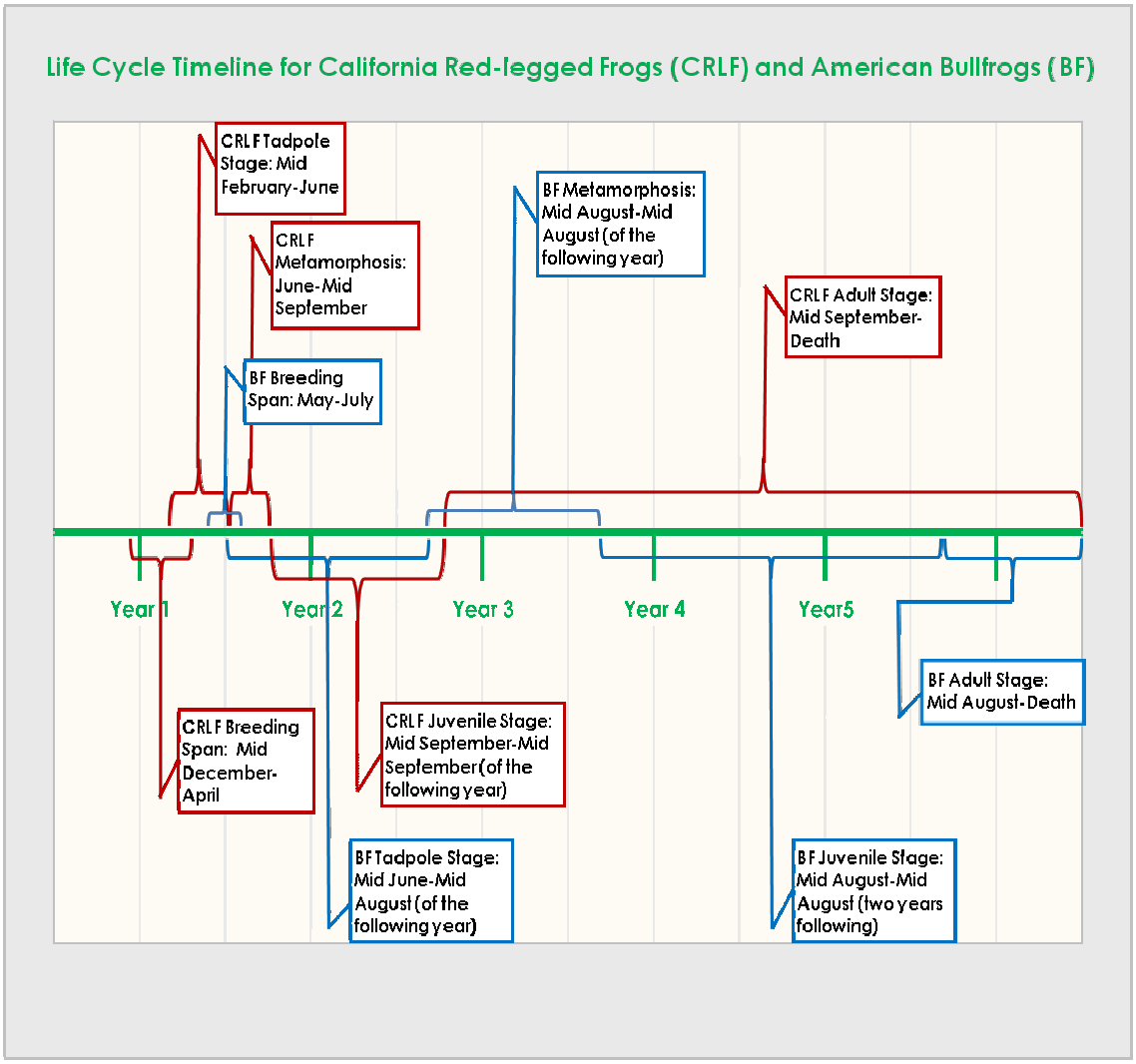


Figure 1: Timeline of a single clutch's development from egg to adult for the California Red-legged Frog and American Bullfrog.

In accounting for all of these changes and additions, we introduce the following notation. We use the letter D (for *draytonii*) to represent our CRLF population, differentiating the stages according to descriptive subscripts. D_T represents the tadpole stage, the juvenile stage is D_J, and the adult stage is D_A. We use the letter C (for *catesbeianus*) to represent the bullfrog population. Since bullfrog tadpoles usually overwinter, we have two tadpole stages C_{T_1} and C_{T_2}. There are also two juvenile stages denoted by C_{J_1} and C_{J_2}. Lastly, the adult bullfrog population is represented by C_A (see Table 1 for parameter names, explanations, and values).

Parameter	Definition	Mean Parameter Values	Reference	Original Parameter/ Calculation
S_1	CRLF tadpole and first year juvenile survivorship	0.00625	Doubledee et al. (2003) Licht (1974)	$P_1 \cdot P_2$
S_2	CRLF second year juvenile survivorship	0.4	Licht (1974)	P_3
S_3	CRLF adult survivorship	0.5	Licht (1974)	P_4
r	CRLF fecundity (eggs/adult)	1,500	Jennings and Hayes (1994)	r
P_1	First year bullfrog tadpole survivorship	0.1	Cecil and Just (1979)	S_0
P_2	Second year bullfrog tadpole survivorship	0.02	Doubledee et al. (2003)	S_1
P_{ft}	Fast track bullfrog tadpole survivorship	0.016	Govindarajulu et al. (2005)	a_{42}
P_3	First year bullfrog juvenile survivorship	0.26	Govindarajulu et al. (2005)	a_{54}
P_4	Second year bullfrog juvenile survivorship	0.32	Doubledee et al. (2003)	S_2
P_5	Adult bullfrog survivorship	0.65	Raney (1940)	S_3
b	Bullfrog fecundity (eggs/adult)	4,000	Bury and Whelan (1984)	b
γ	Intraspecific attack rate on all bullfrog tadpoles	0.02	Doubledee et al. (2003)	γ
μ	Intraspecific attack rate on all bullfrog juveniles	0.05	Doubledee et al. (2003)	μ
η	Intraspecific attack rate on CRLF tadpoles	0.033	Doubledee et al. (2003)	η
Δt	Time step (years)	1	Doubledee et al. (2003)	Δt

Table 1: Symbols and definitions for model parameters. All values were determined empirically via field surveys conducted by each author or collection of authors.

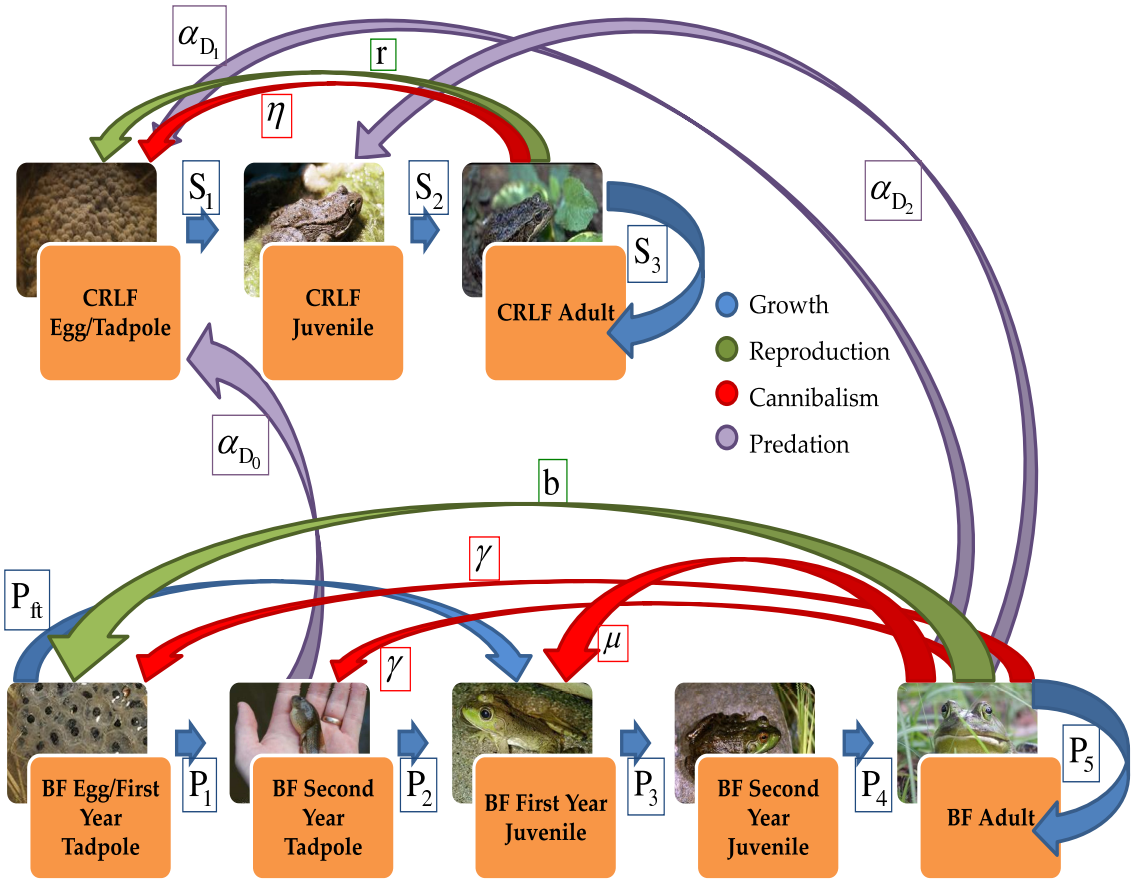


Figure 2: A direction graph showing the life cycles and interactions of our two focal species. Parameter values are described in Table 1.

The fraction of bullfrog tadpoles that survive cannibalism is represented by

$$f_{C_{T_{1,2}}} = e^{-\gamma C_{A(t)} \Delta t} \tag{1}$$

where γ is the attack rate of adult bullfrogs on conspecific tadpoles. The structure of this function allows the survival of these tadpoles to be greater when there are fewer adult frogs

present. Similarly, the proportion of juvenile bullfrogs that survive intraspecific predation is

$$f_{C_{J_{1,2}}} = e^{-\mu C_{A(t)} \Delta t} \tag{2}$$

where μ is the attack rate of adult bullfrogs on conspecific juveniles. The fraction of CRLF tadpoles which survive both interspecific and intraspecific predation is:

$$f_{D_T} = e^{-(\eta D_{A(t)} + \alpha_{D_1} C_{A(t)}) \Delta t}. \tag{3}$$

Finally, the survival of juvenile CRLFs from bullfrog predation is

$$f_{D_J} = e^{-\alpha_{D_2} C_{A(t)} \Delta t} \tag{4}$$

where $\alpha_{D_{1,2}}$ are interspecific attack rates of the bullfrog on CRLF tadpoles and juveniles, respectively.

In the Doubledee et al. (2003) model, there is no explicit interaction between the tadpole stages of the two species. From previous work (Blaustein and Kiesecker, 1998; Kupferberg, 1997; Kiesecker and Blaustein, 1997) we know that this interaction can be quite detrimental to the survival of the larval CRLF since large overwintered bullfrog tadpoles will not hesitate to nibble on CRLF eggs. We model this interaction in the same way that Doubledee et al. (2003) modeled bullfrog adult predation, presented below:

$$f_{D_0} = e^{-\alpha_{D_0} C_{A(t)} \Delta t} \tag{5}$$

where α_{D_0} is the attack rate of bullfrog tadpoles on CRLF tadpoles. Our completed matrix equations for CRLFs and bullfrogs are presented below as equations 6 and 7, respectively.

For ease of understanding, a direction graph which clearly shows all transitions and interactions is provided in Figure 2.

$$\begin{bmatrix} D_T \\ D_J \\ D_A \end{bmatrix}_{t+1} = \overbrace{\begin{bmatrix} 0 & 0 & rS_3 \\ S_1 f_{D_T}(D_A, C_{T_2}, C_A) & 0 & 0 \\ 0 & S_2 f_{D_J}(C_A) & S_3 \end{bmatrix}}^{\mathbf{R}} \cdot \begin{bmatrix} D_T \\ D_J \\ D_A \end{bmatrix}_t , \tag{6}$$

$$\begin{bmatrix} C_{T_1} \\ C_{T_2} \\ C_{J_1} \\ C_{J_2} \\ C_A \end{bmatrix}_{t+1} = \overbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & bP_5 f_{C_{T_1}}(C_A) \\ P_1 f_{C_{T_2}}(C_A) & 0 & 0 & 0 & 0 \\ P_{ft} f_{C_{J_1}}(C_A) & P_2 f_{C_{J_1}}(C_A) & 0 & 0 & 0 \\ 0 & 0 & P_3 & 0 & 0 \\ 0 & 0 & 0 & P_4 & P_5 \end{bmatrix}}^{\mathbf{B}} \cdot \begin{bmatrix} C_{T_1} \\ C_{T_2} \\ C_{J_1} \\ C_{J_2} \\ C_A \end{bmatrix}_t . \tag{7}$$

For simplicity, the unknown parameters α_{D_0}, α_{D_1}, and α_{D_2} are organized in an ordered triplet: (α_{D_0}, α_{D_1}, α_{D_2}). We performed a parameter study on these unknowns and found that depending on the choice of values for the entries of the triplet, the amount of time in which we assume these species can coexist is determined for the system. We do this because it is still unknown to us how long these two species can coexist in this specific habitat. We determined possible ordered triplets for 20, 40, 60, 80, 100, and 200 year coexistence schemes (Figure 3). Figures 4(a), 4(b), and 4(c) show how the CRLF population responds to different coexistence schemes based on different choices for the alpha triplet.

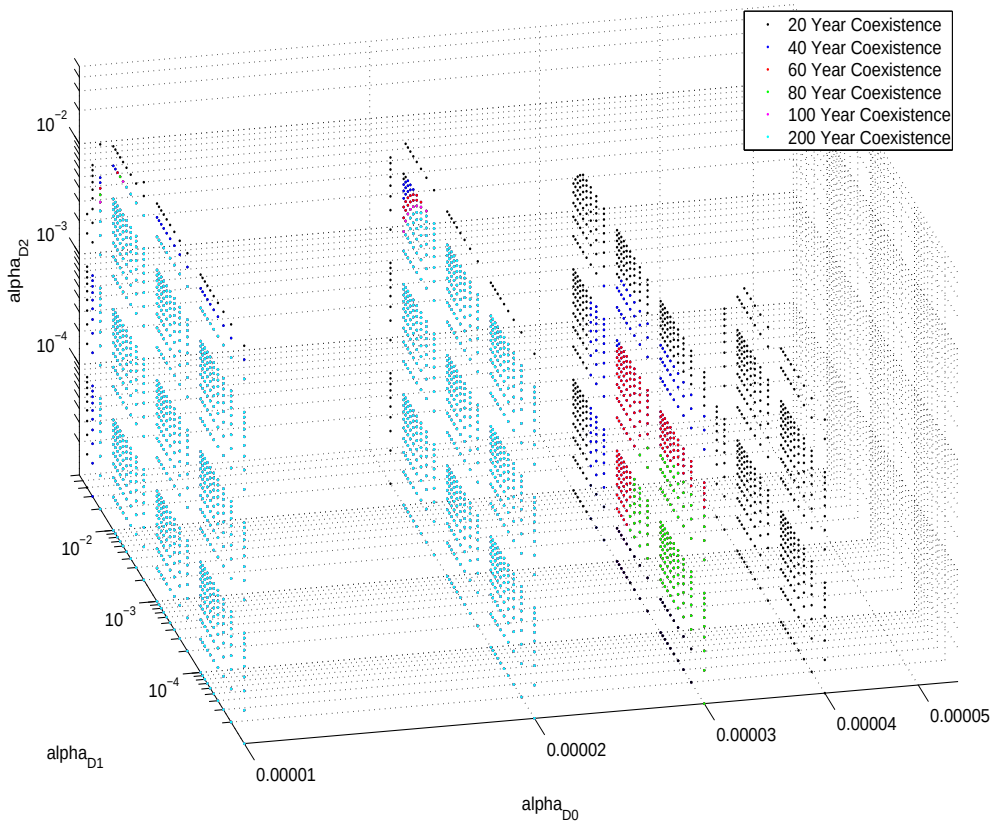
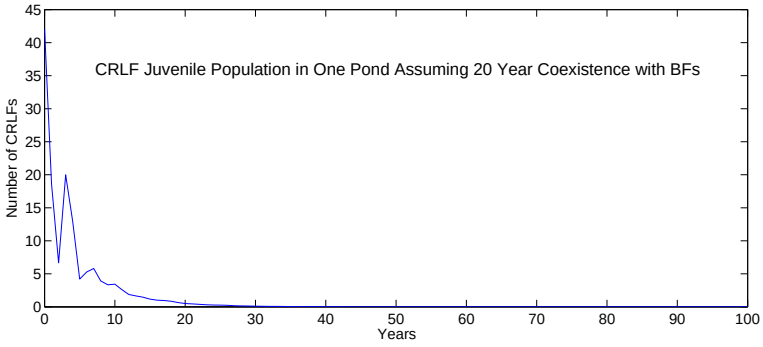


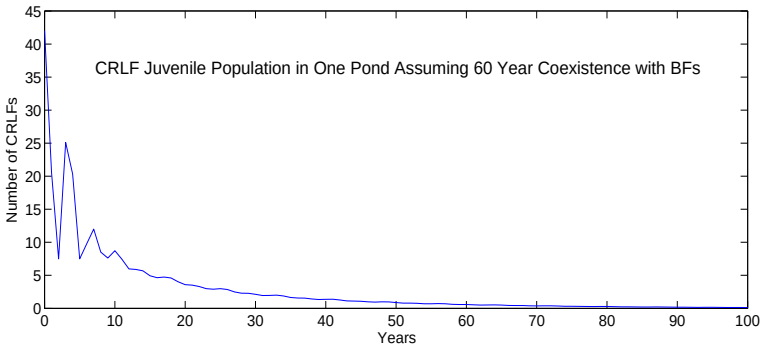
Figure 3: Possible unknown α triplets for various coexistence scenarios. The color assignments represent points that allow that specific coexistence duration. For example, red points represent α triplets that lead to coexistence of more than 40 years but less than or equal to 60 years.

Multiple Pond Model

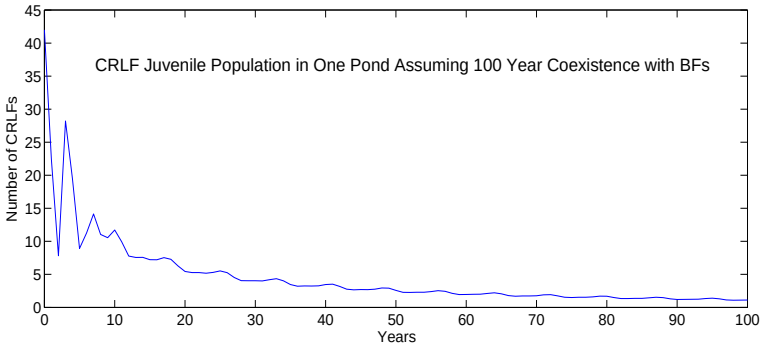
We extend the one pond model to describe movement between ponds by incorporating immigration and emigration terms that allow for movement of animals in a spatial configuration of discrete ponds. We assume there are specific influences governing the movement of these animals, namely proximity of the animal to other ponds, stage, and habitat quality. We incorporate proximity information by using dispersal data for CRLFs at the Point Reyes



(a) $(\alpha_{D_0}, \alpha_{D_1}, \alpha_{D_2}) = (0.00004, 0.003, 0.0001)$



(b) $(\alpha_{D_0}, \alpha_{D_1}, \alpha_{D_2}) = (0.00003, 0.001, 0.0008)$



(c) $(\alpha_{D_0}, \alpha_{D_1}, \alpha_{D_2}) = (0.00002, 0.01, 0.003)$

Figure 4: Examples of the choice of alpha triplet determining the length of time in which the species coexist. Here we only show the juvenile CRLF population (for simplicity) over various durations of coexistence: (a) 20 years (b) 60 years (c) 100 years.

National Seashore in Marin County, California (Fellers and Kleeman, 2007, map shown in Figure 5, data shown in Figure 6). These data allow us to calculate proportions of frogs that move between ponds. Our movement rule encompasses an influence due to habitat quality, uniquely designed for both species. Also, the stage of the animal plays a distinct role in the choice of movement direction, given that there is a choice in moving to several connected ponds. We define 'connected' to mean within a predetermined dispersal range.

We assume that only the juvenile frogs in the simulation move between ponds. We do this for two reasons: 1) juveniles have been observed to be 'the [main] dispersers' among many ranid species (Lannoo, 2005) and 2) CRLFs and bullfrogs both exhibit high philopatry (the tendency of individuals to return to their initial breeding pond to breed in the future) (Fellers and Kleeman, 2007; Stinner et al., 1994). However, we recognize that there are shortcomings with these assumptions. The data set (Fellers and Kleeman, 2007) was collected via radio telemetry and due to the size of the transmitter, only adult frogs were large enough to be fitted. Since juvenile CRLFs were not included in the study, we were left to assume that juveniles move in a similar fashion as their adult counterparts. Juveniles would probably move shorter distances than adults, but as the telemetry data are likely to be an underestimate of this species dispersal capabilities, we contend that our use of this data is reasonable. Furthermore, although the Fellers and Kleeman (2007) data show that adult frogs indeed move, sometimes substantial distances, we assume these movements are made to satisfy needs outside of the breeding season (to find food, basking sites, etc.). Because of the suggestions of high philopatry, we assume that adult frog movements outside of the census (i.e., during the breeding season) and are therefore inconsequential for our purposes. As far as the simulation is conserned, the adults appear not to move at all since we are only interested with where they end up at the time of the census, in which we assume they always return to their initial breeding pond. The purpose of the movement rule then becomes a

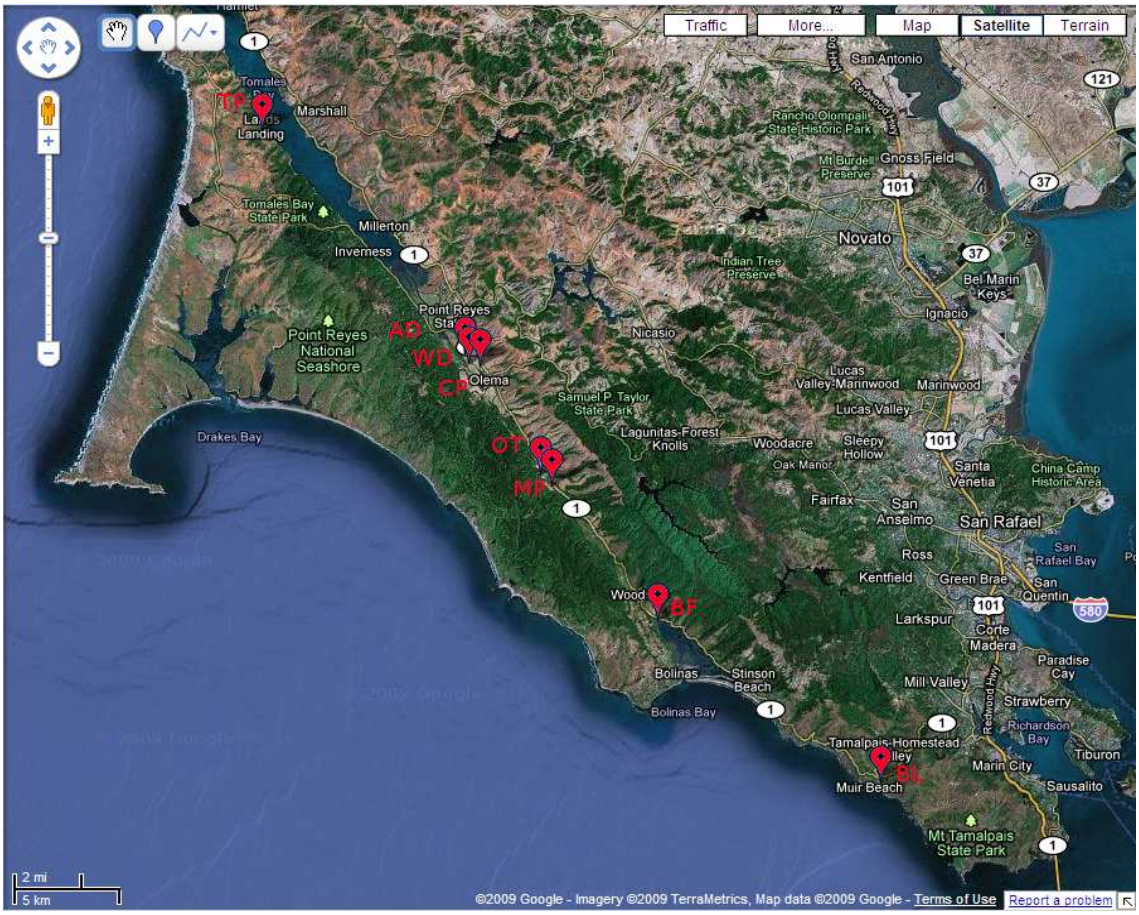


Figure 5: Map of Point Reyes National Seashore and Golden Gate National Recreation Area of Marin County, California. Marked areas represent study sites where dispersal data for the California Red-legged Frog were collected by Fellers and Kleeman (2007).

mechanism for juvenile frogs to find a place where they will breed in the future.

The dispersal data (Figure 6) collectively exhibits the shape of a decaying exponential function. Due to this observation, and in order to use these data to extract proportions of moving frogs based on distance, we use gamma distributions. A gamma distribution can exhibit a variety of shapes, one of which resembles a decaying exponential function not unlike the shape of our dispersal data. We fit these data to a gamma distribution using maximum likelihood (Rice, 1995). This was done using a Matlab code called *gamfit.m*

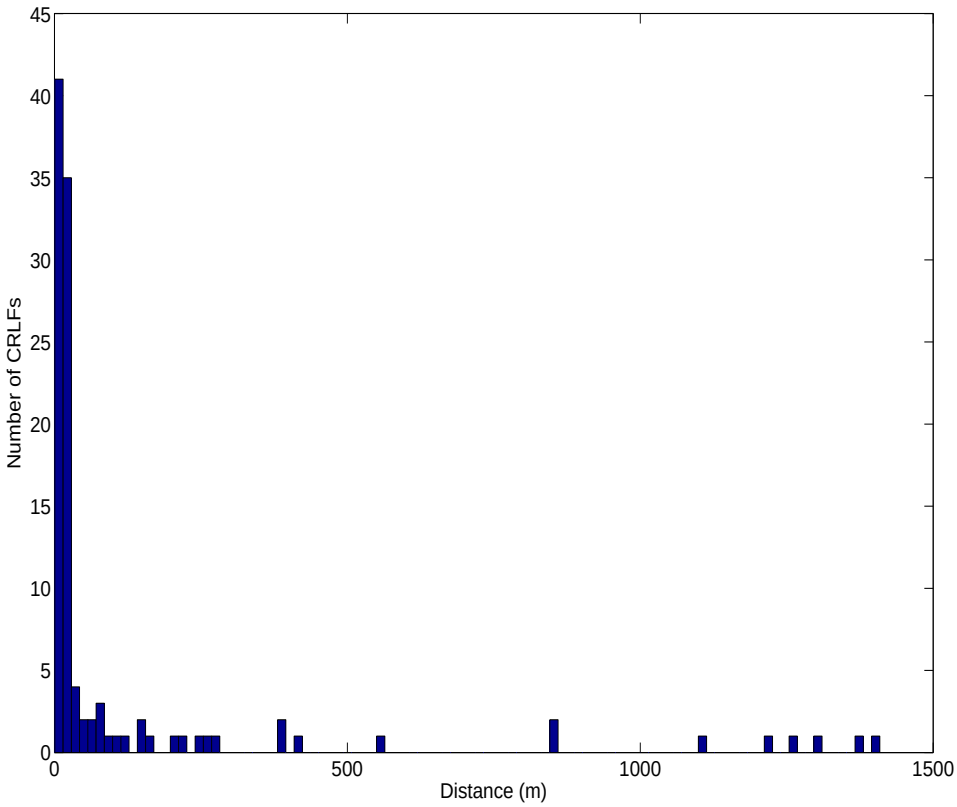


Figure 6: Histogram of dispersal distance data for CRLFs along the Point Reyes National Seashore and Golden Gate National Recreation Area of Marin County, California, collected by Fellers and Kleeman (2007).

taken from Matlab's statistics package. We simply fed the dispersal data into the routine and it produced parameters α and β associated with gamma distributions. The α parameter describes the shape of the curve produced (by the gamma distribution) while the β parameter determines the scale. Assuming similar dispersal behavior, the α parameters are the same for each species, while the β parameter is different from the calculated value for the bullfrog gamma distribution, allowing for a 'longer tail'. We must do this because dispersal data of the same ilk for the bullfrog do not exist to our knowledge. An example

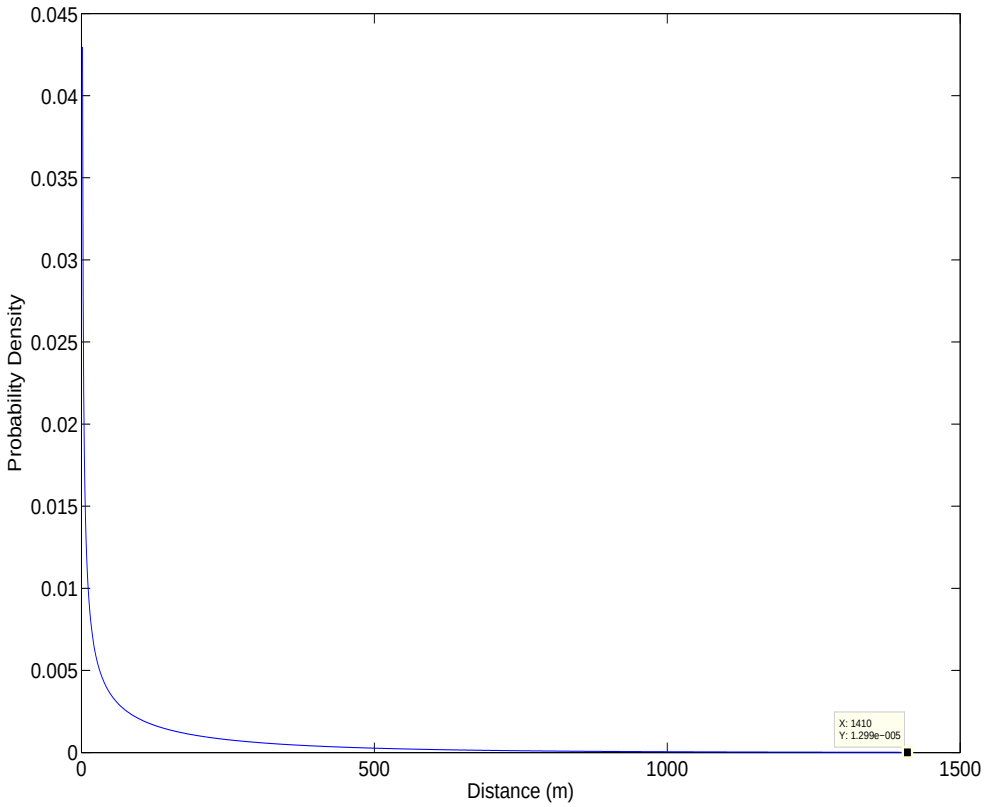


Figure 7: A gamma distribution that emulates the CRLF dispersal distance data histogram. Note that the largest distance value is very close to the maximum distance of 1409 meters recorded in the Fellers and Kleeman (2007) data set.

of a gamma distributed curve created for CRLFs is displayed in figure 7, and one for the bullfrog is shown in figure 8.

The gamma distribution is used to calculate the proportion of a population that is able to move a given distance. For example to caclulate the proportion of the population that can at to 200 meters, we find the area under the gamma distribution to the distance of 200 meters. That area represents the proportion that can move *up to* 200 meters, so in order to find the proportion that can move 200 meters or more, we calculate the compliment of

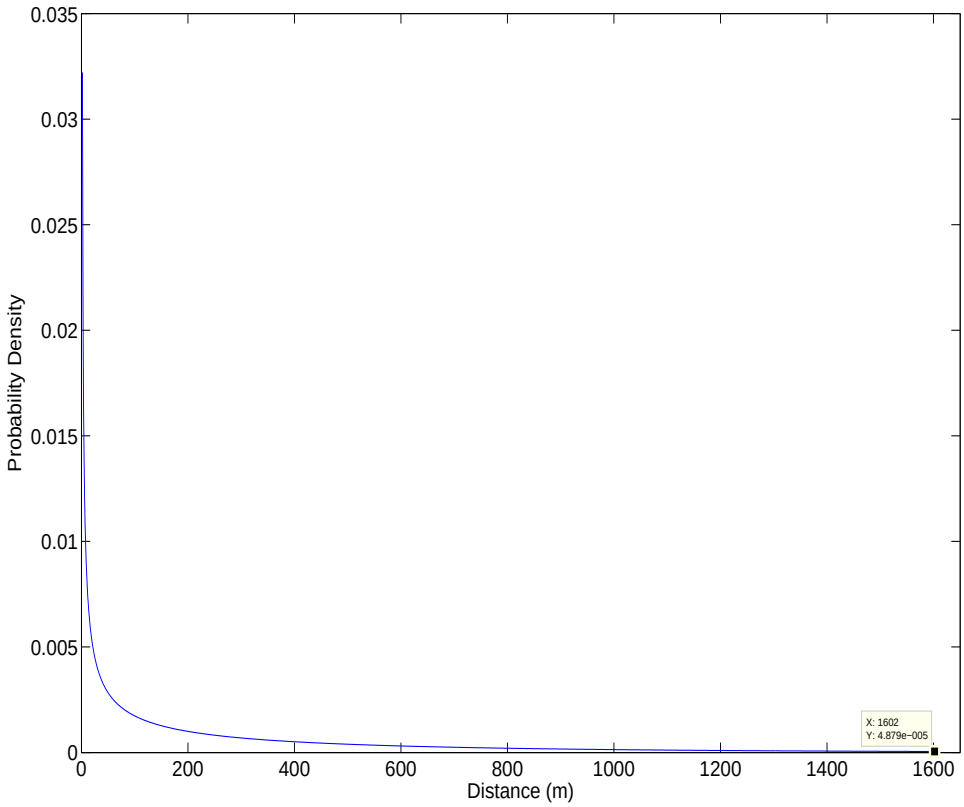


Figure 8: A gamma distribution, based on the shape of the CRLF gamma distribution, that estimates bullfrog movement. Note that the largest distance value is very close to the maximum distance of 1600 meters recorded in the work of Marsh and Trenham (2001).

the aformentioned proportion (i.e., 1 - proportion moving up to 200 meters). An illistration showing the area that represents the proportion caclulated for this specific example is shown in Figure 9. For each distance, there is associated with it a proportion of the population moving from pond i to pond j (called $r_{i,j}$) which we store in the matrix R.

In order to find the fraction of populations dispersing between and two ponds, we used distances between the eight sites that were part of the Fellers and Kleeman (2007) study. The straight line distances between each of the were measured in meters using Google

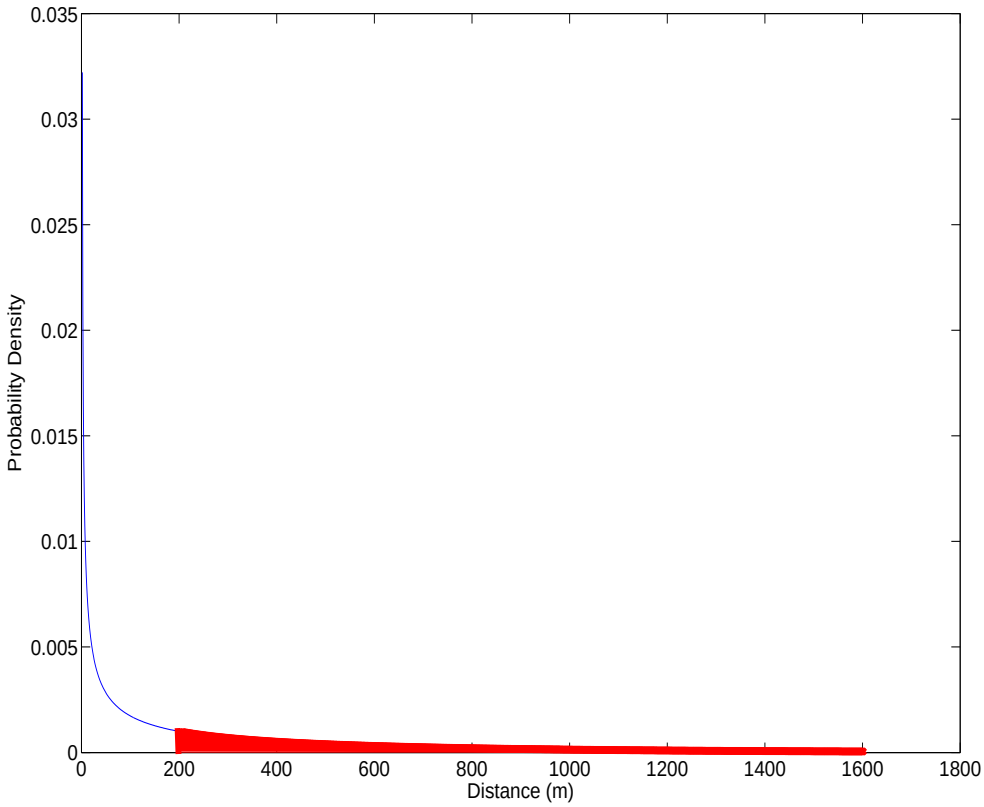


Figure 9: An example of calculating the proportion of dispersing frogs using a gamma distribution. The red area the proportion of individuals that could travel a distance of at least 200 meters. In this case, 38.76% of the bullfrogs in this particular pond are able to move that specified distance.

Maps (Figure 5) and stored in a distance matrix M (Equation 8). In this matrix the row number is the current pond location and column number is the destination pond location. For instance, the distances between pond eight (pond BL) (see Table 2 for pond numbers) and any other pond can be found in the eighth row. The first column entry of the eighth row is the distance between ponds one (pond TP) and eight (pond BL). Similarly the distance between ponds eight and two (pond AD) is the second column entry of the eighth row and

would be denoted $m_{8,2}$. Note that the diagonal values of M are zero, since every pond has a zero distance from itself ($m_{i,i} = 0 | i = 1, 2, \ldots, 8$), and the matrix is symmetric, i.e., $m_{i,j} = m_{j,i}$.

Pond Name	Pond Number	Hydroperiod
TP	1	Seasonal Seep and Ditch
AD	2	Seasonal Pond
WD	3	Permanent Pond
CP	4	Permanent Pond
OT	5	Permanent Pond
MP	6	Seasonal Pond
BF	7	Seasonal Pond
BL	8	Permanent Marsh

Table 2: Numeric assignments for our eight study ponds. Names and hydroperiods taken from Fellers and Kleeman (2007).

$$M = \begin{bmatrix} 0 & 14740 & 15240 & 15640 & 21610 & 22400 & 30830 & 43870 \\ 14740 & 0 & 519.38 & 885.14 & 6950 & 7720 & 16130 & 29130 \\ 15240 & 519.38 & 0 & 541.12 & 6440 & 7210 & 15610 & 28650 \\ 15640 & 885.14 & 541.12 & 0 & 6040 & 6840 & 15210 & 28210 \\ 21610 & 6950 & 6440 & 6040 & 0 & 677.97 & 9110 & 22270 \\ 22400 & 7720 & 7210 & 6840 & 677.97 & 0 & 8400 & 21580 \\ 30830 & 16130 & 15610 & 15210 & 9110 & 8400 & 0 & 13340 \\ 43870 & 29130 & 28650 & 28210 & 22270 & 21580 & 13340 & 0 \end{bmatrix} \quad (8)$$

We have two distinct ways of creating matrix R: deterministically and stochastically. In the deterministic version, we assume that the population sizes of each species among the ponds have no effect on the pattern/frequency of dispersal between ponds. In this case, the

gamma distribution used is exactly fitted to the data. Only one gamma distribution for each species (Figures 7 and 8) will be used to create matrices R_D and R_C, respectively. Note that each R matrix is symmetric (i.e., the probability of movement is the same whether individuals are moving from pond i to pond j or from pond j to pond i).

In the stochastic version, we assume that the population sizes of each pond influence the rate of movement between the ponds. For this simulation, a distinct set of n_i (population size at pond i, rounded to the nearest integer) gamma distributed random numbers are generated. In this case, a larger population means that there is an increased likelihood of individuals dispersing further distances whereas in the deterministic model, the fraction of the population that is able to move a given distance if fixed, reguardless of the populaton size. Since each proportion is calculated from distinct collections of gamma distributed random numbers, the matrix R for the stochastic version may be asymmetric and unique for every time step.

With the absence of concrete data that clearly support a connection between habitat quality and survival, we assume that hydroperiod (length of time, or seasonality, that water is present over the surface of the landscape) will represent the influences of survival due to habitat quality for the bullfrog. The hydroperiod for each pond is provided in the study from which we take our dispersal distance data (Fellers and Kleeman, 2007, Table 2). We do this because although there are other influences which describe habitat quality (e.g., pond temperature, depth, surface area, amount of emergent vegetation, etc.) these attributes are so highly correlated with the hydroperiod of the pond that we assume they are incorporated within this characteristic. We allow for two possible hydroperiods in our study: permanent and seasonal. We assume that overwintered bullfrog tadpoles can only survive in permanent ponds. We reflect this phenomenon in our model by initializing all permanent ponds with null bullfrog populations and eliminating any overwintered tadpoles

which have been deposited in seasonal ponds by immigrant breeding adults every year. We further assume that CRLFs can survive in either permanent or seasonal habitats.

Hydroperiod also effects the movement of the bullfrog. We assume second year juvenile bullfrogs avoid dispersing to seasonal ponds considering that they had the previous year to become knowledgeable about their landscape (see equation 9). However, first year juvenile bullfrogs do not have this information and therefore blindly choose to move between any pond within their dispersal range (see figure 10).

$$\delta_{ij}^{C,quality} = \begin{cases} 1 & \text{if } j \text{ is permanent} \\ 0 & \text{if } j \text{ is seasonal} \end{cases} \tag{9}$$

We have also implemented an indicator of habitat quality for the CRLF as well. When juvenile CRLFs emerge from their natal pond, they have no information available to them than what they see at their current location. We thus assume that the density of the adult bullfrog population at that pond influences whether or not the juvenile CRLFs will move out of the pond. We designed this probability so that the higher the adult bullfrog density, the more likely CRLFs are to leave that pond for, hopefully, a pond with a smaller population of bullfrogs (see equation 10). Thus, the probability of movement of CRLFs out of pond i to any connected pond based on the adult bullfrog density is as follows:

$$\delta_i^{D,quality} = 1 - \frac{1}{C_{A_i}}. \tag{10}$$

We assume the two stages of juvenile frogs present in our model have distinct modes of choosing between several connected ponds. The first year juvenile bullfrogs (and the sole juvenile stage for CRLFs) have just emerged out of the pond, completing their metamorphosis. We thus assume they have no directional preference. Should individuals of this

stage choose to move out of their current pond, we assume it is equally probable for them to move toward any given connected pond (for clarification, see figure 10). On the other hand, second year juvenile bullfrogs have had a chance to absorb information about the landscape (i.e., the distances between all connected ponds in the area). We then assume that the distances between the connected ponds play a role in the choice they make on which pond they decide to move to, if they move at all (see figure 11).

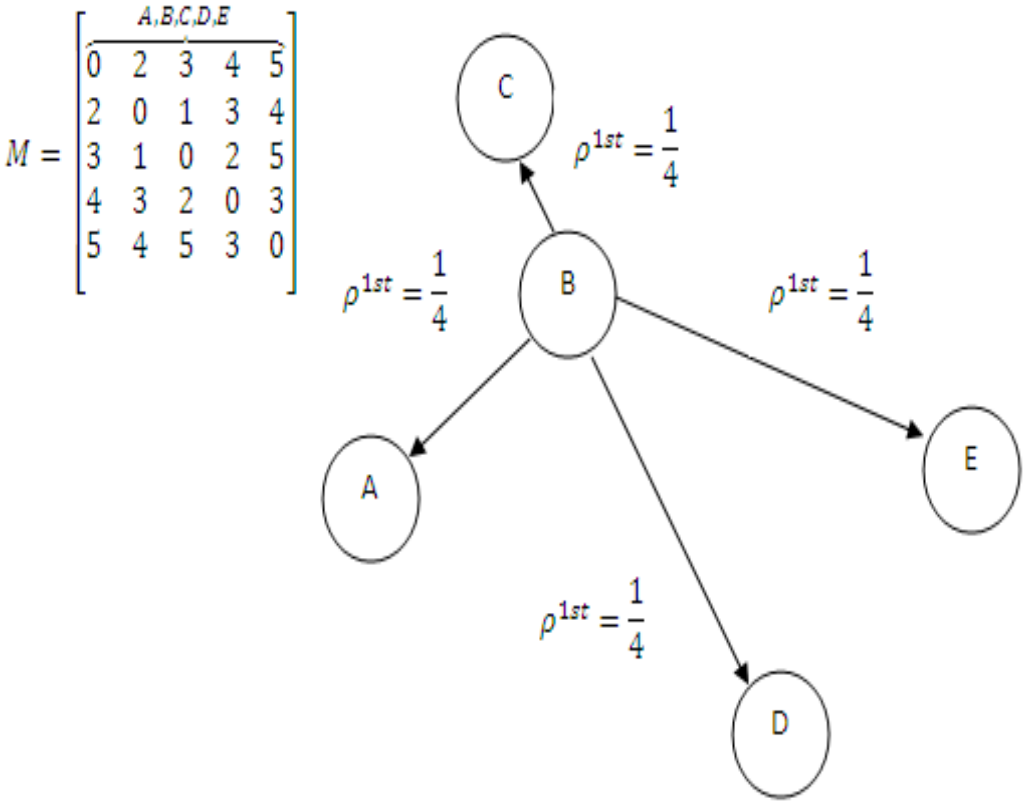


Figure 10: Example of juvenile CRLF and first year bullfrog movement choice probabilities. Note that all choice probabilities ρ^{1st} are the same.

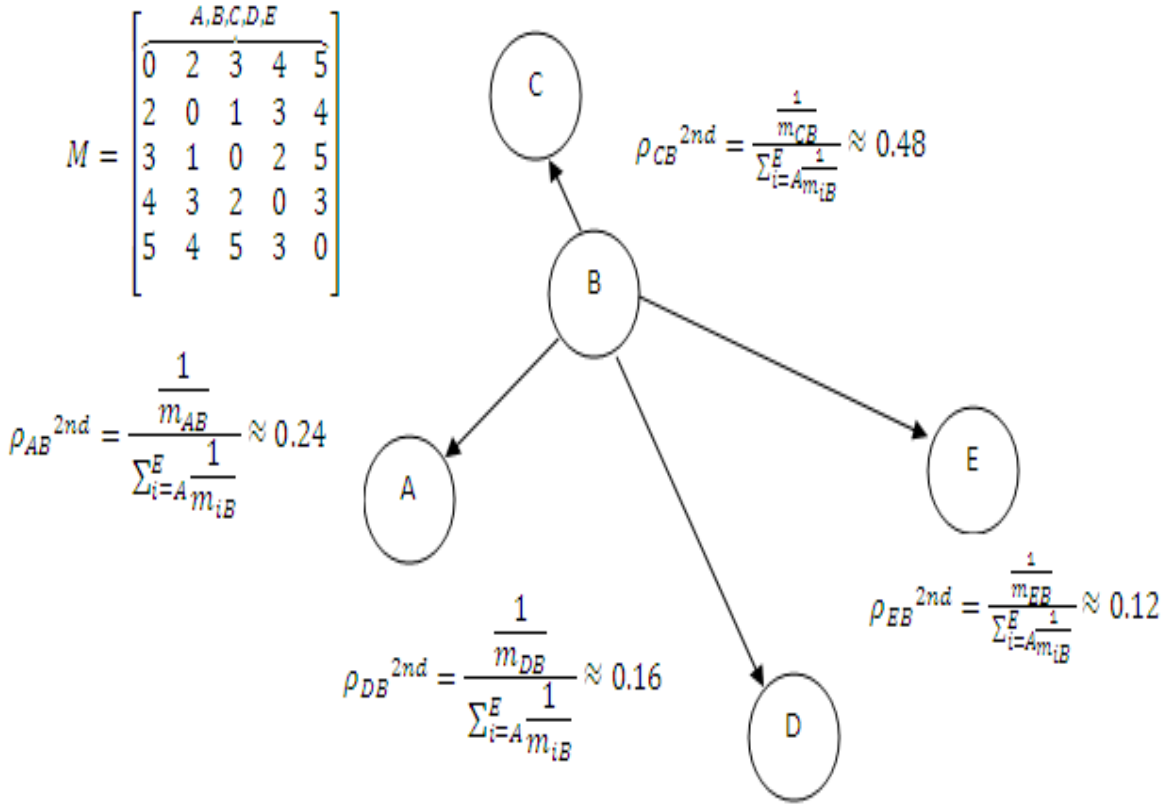


Figure 11: Example of second year bullfrog movement choice probabilities. The pair of ponds which encompasses the shortest distance (in this case, the distance between ponds B and C) will have associated with is the highest choice probability, ρ_{CB}^{2nd}. In this illustration, we are assuming that all ponds are permanent so our second year juveniles would not avoid any ponds via the habitat quality component of our movement rule.

We now have a new model which has its foundation from our one pond model, only now immigration and emigration terms are added to and subtracted from it, respectively.

$$\begin{bmatrix} D_{T_i} \\ D_{J_i} \\ D_{A_i} \end{bmatrix}_{t+1} = \overbrace{\begin{bmatrix} 0 & 0 & rS_3 \\ S_1 f_{D_{T_i}}(D_{A_i}, C_{T2_i}, C_{A_i}) & 0 & 0 \\ 0 & S_2 f_{D_{J_i}}(C_{A_i}) & S_3 \end{bmatrix}}^{\mathbf{R}} \cdot \begin{bmatrix} D_{T_i} \\ D_{J_i} \\ D_{A_i} \end{bmatrix}_t - \begin{bmatrix} 0 \\ r_{ij} \cdot \delta_i^{D,quality} \cdot \rho^{1st} \cdot D_{J_i} \\ 0 \end{bmatrix}_t + \begin{bmatrix} 0 \\ 0 \\ \sum_{j \neq i} r_{ji} \cdot \delta_j^{D,quality} \cdot \rho^{1st} \cdot D_{J_j} \end{bmatrix}_t, \tag{11}$$

$$\begin{bmatrix} C_{T1_i} \\ C_{T2_i} \\ C_{J1_i} \\ C_{J2_i} \\ C_{A_i} \end{bmatrix}_{t+1} = \overbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & bP_5 f_{C_{T1_i}}(C_{A_i}) \\ P_1 f_{C_{T2_i}}(C_{A_i}) & 0 & 0 & 0 & 0 \\ P_{ft} f_{C_{J1_i}}(C_{A_i}) & P_2 f_{C_{J1_i}}(C_{A_i}) & 0 & 0 & 0 \\ 0 & 0 & P_3 & 0 & 0 \\ 0 & 0 & 0 & P_4 & P_5 \end{bmatrix}}^{\mathbf{B}} \cdot \begin{bmatrix} C_{T1_i} \\ C_{T2_i} \\ C_{J1_i} \\ C_{J2_i} \\ C_{A_i} \end{bmatrix}_t - \begin{bmatrix} 0 \\ 0 \\ r_{ij} \cdot \rho^{1st} \cdot C_{J1_i} \\ r_{ij} \cdot \delta_{ij}^{C,quality} \cdot \rho_{ij}^{2nd} \cdot C_{J2_i} \\ 0 \end{bmatrix}_t + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \sum_{j \neq i} r_{ji} \cdot \rho^{1st} \cdot C_{J1_j} \\ \sum_{j \neq i} r_{ji} \cdot \delta_{ij}^{C,quality} \cdot \rho_{ji}^{2nd} \cdot C_{J2_j} \end{bmatrix}_t \cdot \tag{12}$$

Management

Since our choice of coexistence scheme assumes the CRLF cannot continue to survive in a single pond without management, we can use our model to investigate the efficacy of various management strategies for the benefit of this threatened species. We recognize many different management strategies to deal with nuisance bullfrogs (Govindarajulu et al., 2005; Doubledee et al., 2003; Moyle, 1973). We entertain three methods of management of bullfrogs: tadpole focused, metamorph focused, and juvenile/adult focused management. These strategies are implemented in the model by simply reducing the survival rates associated with the stages being targeted for eradication by increments of 5%. For instance, when we implement the tadpole focused scenario, the survival rates of the pond bound stages of bullfrog are adjusted to whichever percentage of removal we like.

We ran our simulations to 200 years implementing a specific strategy every year, and averaged the population sizes from only the last 100 years in order to exclude any transient behavior present in the beginning of the simulation run. We compared the average population sizes and variances over several, equally lengthy ranges of time (100-200 years, 150-250 years, 200-300 years, etc.) and found the differences between the average population sizes and variances of all time ranges investigated to be insignificant, thus we kept our time frame between 100-200 years. The populations described in all figures for the Results section will be limited to the CRLF juvenile population, for the sake of brevity.

Since projects may be limited by funding, we investigate the effect that skipping management from one to 15 years in a row has on the average juvenile CRLF population. We assume that every year the management is implemented, it is done at a fixed rate of 100% eradication for whichever stage of bullfrog is the focus of the effort. Again, we run the simulation for 200 years and average only the later 100 years.

RESULTS

Deterministic Model

We eliminated ponds TP and BF from our results because they are disconnected (i.e., beyond dispersal range) seasonal ponds. These characteristics made the ponds devoid of bullfrogs, since we do not initialize this species in seasonal ponds and the disconnection made colonization impossible in our simulation. Furthermore, the fact that these ponds are not connected to any other made for a constant time series for the bullfrog while the CRLF reached a steady state. Although this is an important result, showing that CRLFs can flourish when bullfrogs are absent, it only adds unnecessary clutter to our results.

The bullfrog populations over all stages and locations reach a steady state by about the 20^{th} year (Figure 12). Ponds AD and MP are seasonal, and this seasonality constantly wipes out the second year bullfrog tadpole population and thus lead to smaller bullfrog populations overall. However, the first year juvenile stage at those seasonal ponds actually have larger populations when compared to permanent ponds. We attribute this to the effect of nearby permanent ponds (WD, CP, and OT) feeding relatively high volumes of first year juvenile frogs into these seasonal ponds, since they do not yet discriminate these ponds by their hydroperiod when deciding where to move.

On the other hand, the CRLF populations settle at very low population sizes in the permanent ponds, and even die out by year sixty in pond BL (Figure 13). We are able to control this phenomenon by choosing our alpha triplet (described in Methods) to be $(\alpha_{M_0}, \alpha_{M_1}, \alpha_{M_2}) = (0.00003, 0.001, 0.0008)$ which forces a 60 year coexistence scheme. For the sake of consistency and brevity, we will keep the alpha triplet at this value, assuming a 60 year coexistence between species, for the remainder of this work, unless specified otherwise. Since BL is a disconnected permanent pond, the resulting time series for the

CRLF has exactly the same dynamics as the single pond simulation shown in figure 4(b). Due to this observation, we determined that the other permanent ponds are able to have persistent CRLF populations due to dispersal from connected seasonal ponds, which act as a refuge for this species.

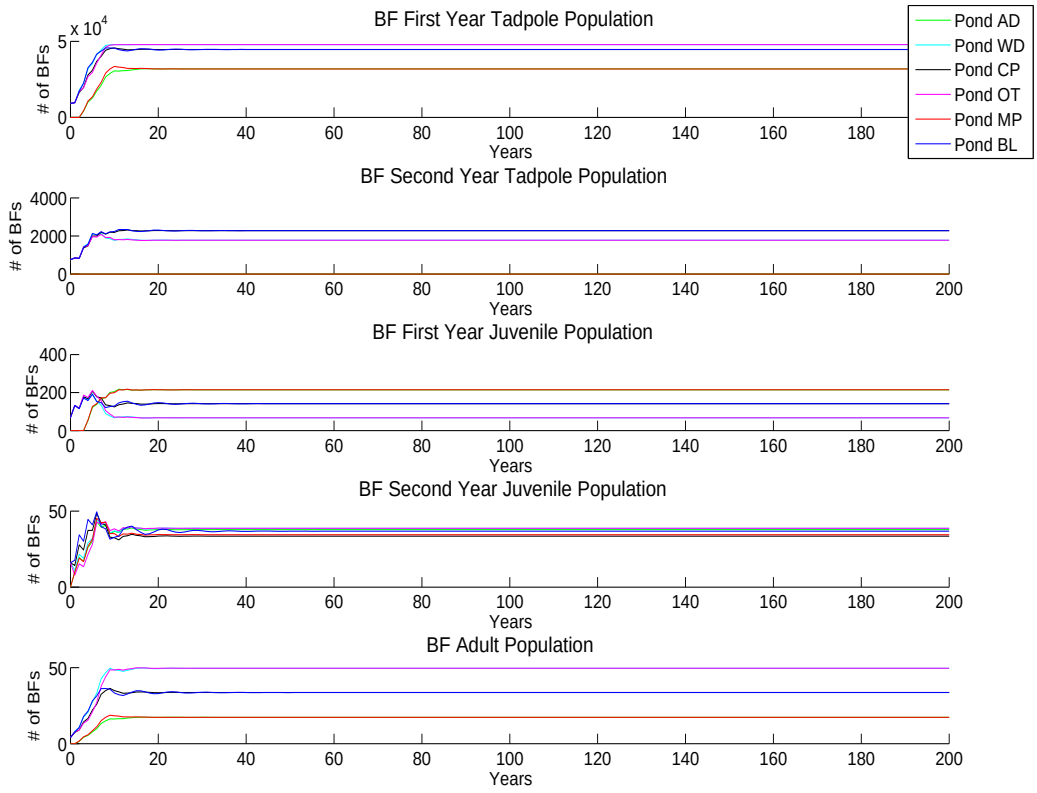


Figure 12: Deterministic simulation result for the bullfrog.

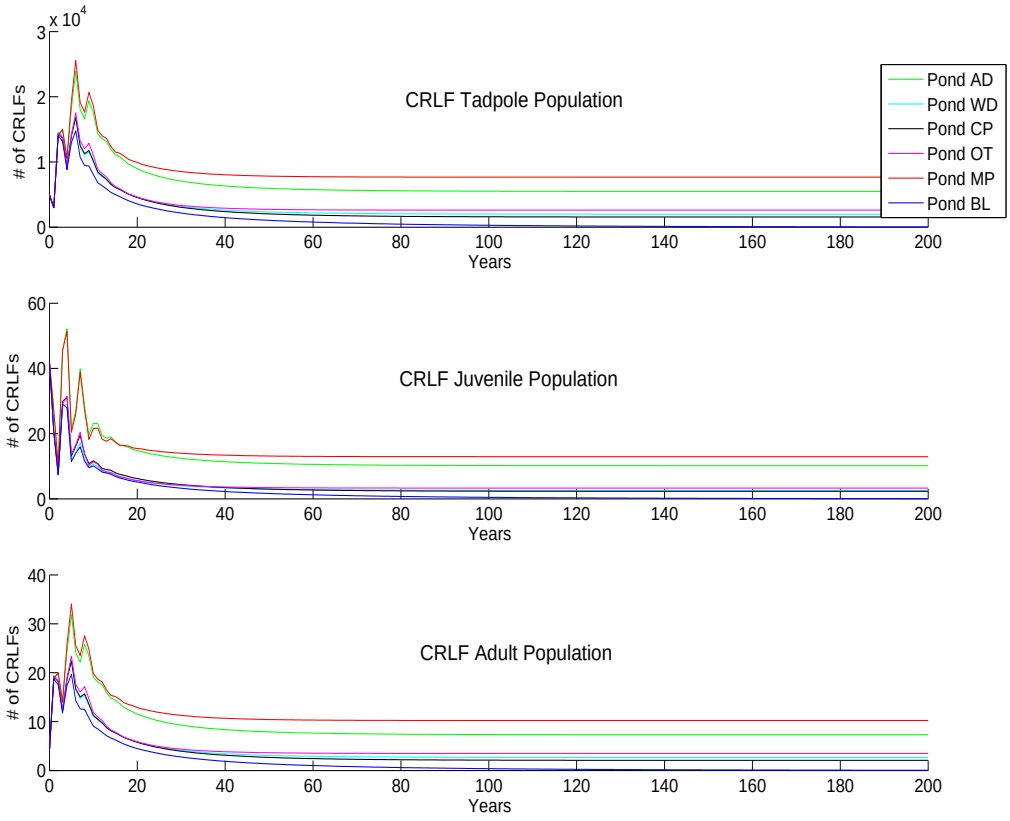


Figure 13: Deterministic simulation result for the CRLF. Assuming a 60 year coexistence in a single pond, we see how CRLFs carry on for the first 200 years of the multiple pond simulation.

Stochastic Model

In this version of the model, we assume the present population size of the pond aides in determining the proportion of moving individuals out of said pond. This allows the gamma distributed histograms from which we calculate our proportions of moving individuals to change with time. Figures 14 and 15 show a 200 year time series of both species as they interact within our six ponds. Notice that the behavior of the stochastic version is qualita-

tively similar to the deterministic version, and in the case of pond BL, the two versions are exactly the same since no movement occurs in or out of that pond. Furthermore, the population sizes of adult and juvenile CRLFs in ponds AD and MP (our two seasonal ponds that act as refuge for CRLFs) are higher in the stochastic version than the deterministic version. The stochastic version has the advantage of allowing for variability in the volumes of moving individuals with time, which is closer to what we would expect in the natural world. For this reason, we use the stochastic version of the model for the remainder of our results, which are composed of all our management scenario simulations.

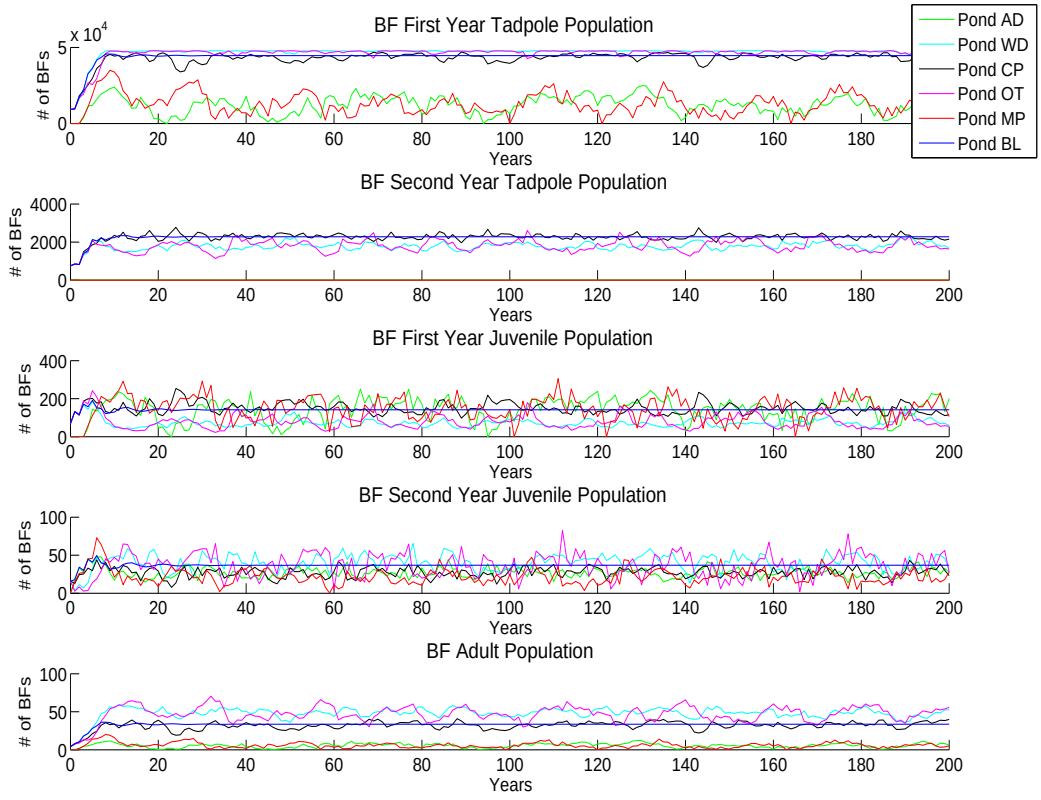


Figure 14: Stochastic simulation result for the bullfrog.

Figure 15: Stochastic simulation result for the CRLF.

Management

For Figures 16, 17, and 18 we see what the effect of removing $0-100\%$ (at 5% increments) of bullfrog tadpoles, metamorphs, and juveniles/adults, respectively, has for the average juvenile CRLF population (with error bars) within each pond. These results illuminate the amount of effort required to effectively manage each specific pond. Figures 19, 20 and 21 show the effect on the average juvenile CRLF population of implementing our strategies full force (100% eradication) every year, every other year, every two years, etc., up to 15

years between management events. Given that funding is not always available from year to year, these results allow us to see whether each management strategy has any lasting effects.

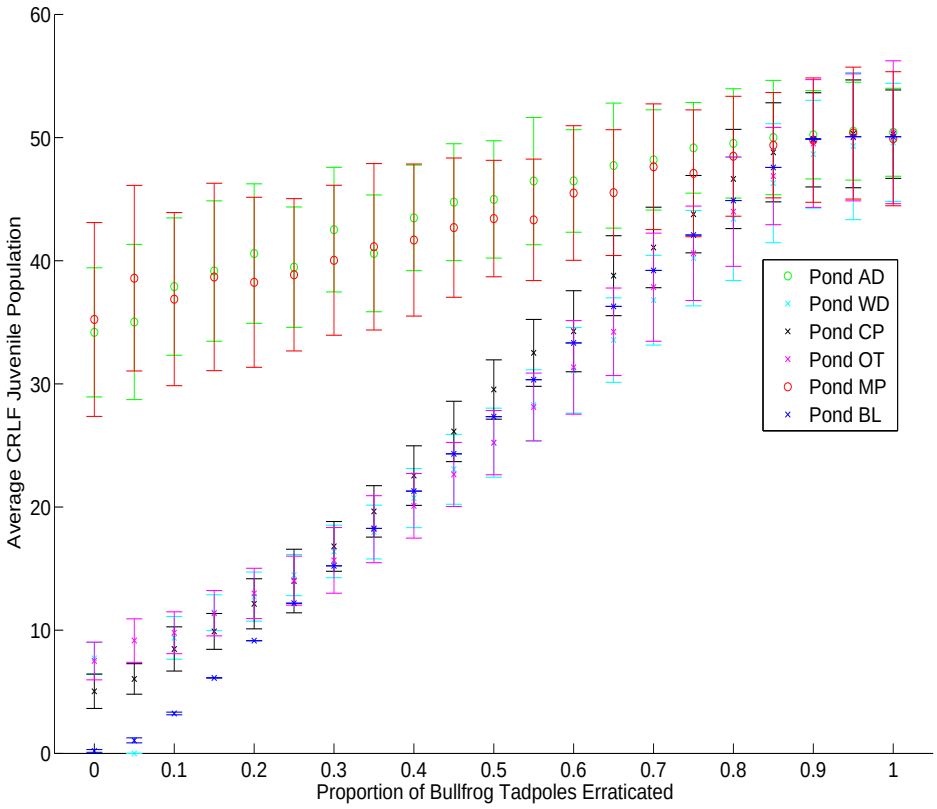


Figure 16: Bullfrog tadpole focused eradication over various proportions of removal. Seasonal ponds are relatively unaffected by bullfrog management. In order to manage the permanent ponds effectively (i.e., have their population sizes match those of seasonal ponds when 0% of bullfrog tadpoles are removed) at least 75% of bullfrog tadpoles must be eliminated every year. Note that for all remaining figures, the seasonal ponds' averages are denoted by an 'o' while the permanent ponds' averages are marked with an 'x'.

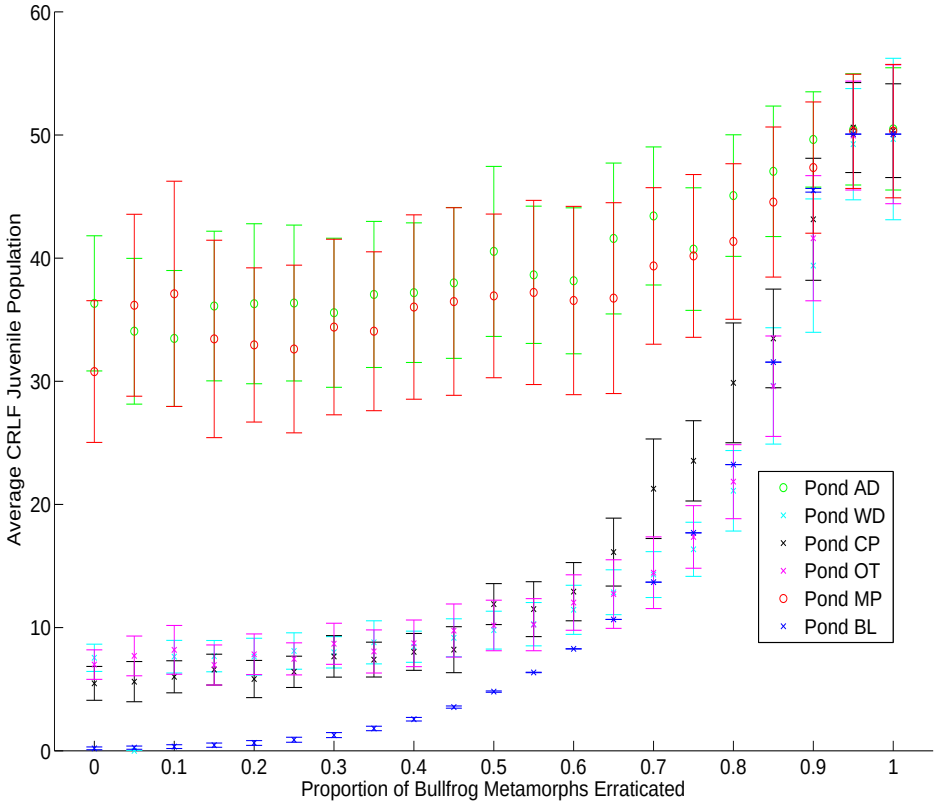


Figure 17: Bullfrog metamorph focused eradication over various proportions of removal. Seasonal ponds need not be managed. This type of management requires at least 90% removal of newly metamorphed bullfrogs from permanent ponds every year in order to sustain healthy CRLF populations. Note that for all remaining figures, the seasonal ponds' averages are denoted by an 'o' while the permanent ponds' averages are marked with an 'x'.

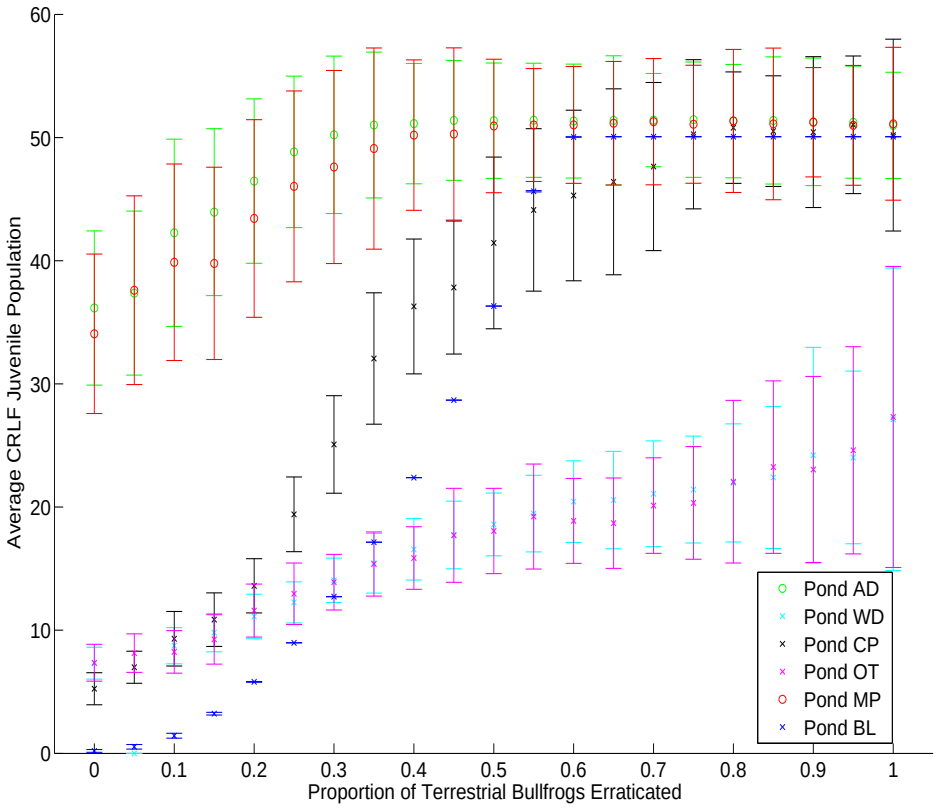


Figure 18: Juvenile/Adult bullfrog focused eradication over various proportions of removal. Seasonal ponds act as a refuge for CRLFs and do not require management. Permanent ponds WD and OT cannot be managed effectively via this strategy, even when 100% of juvenile and adult bullfrogs are taken. However, permanent ponds CP and BL can maintain healthy CRLF populations so long as at least 50% of terrestrial bullfrogs are removed. Note that for all remaining figures, the seasonal ponds' averages are denoted by an 'o' while the permanent ponds' averages are marked with an 'x'.

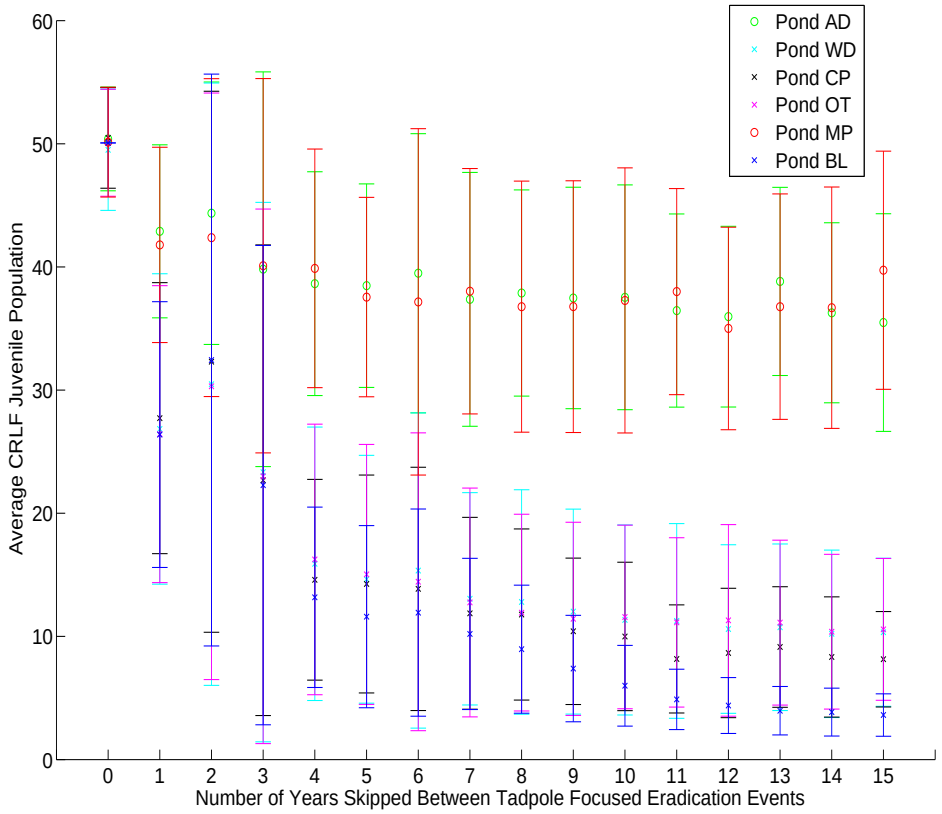


Figure 19: Skipping years between bullfrog tadpole focused eradication efforts. Undoubtedly, continuous management has the best result for the average juvenile CRLF populations of any pond. A clear divergence between average juvenile CRLF population sizes among seasonal and permanent ponds occurs when skipping three or more years in a row between management applications. Note that for all remaining figures, the seasonal ponds' averages are denoted by an 'o' while the permanent ponds' averages are marked with an 'x'.

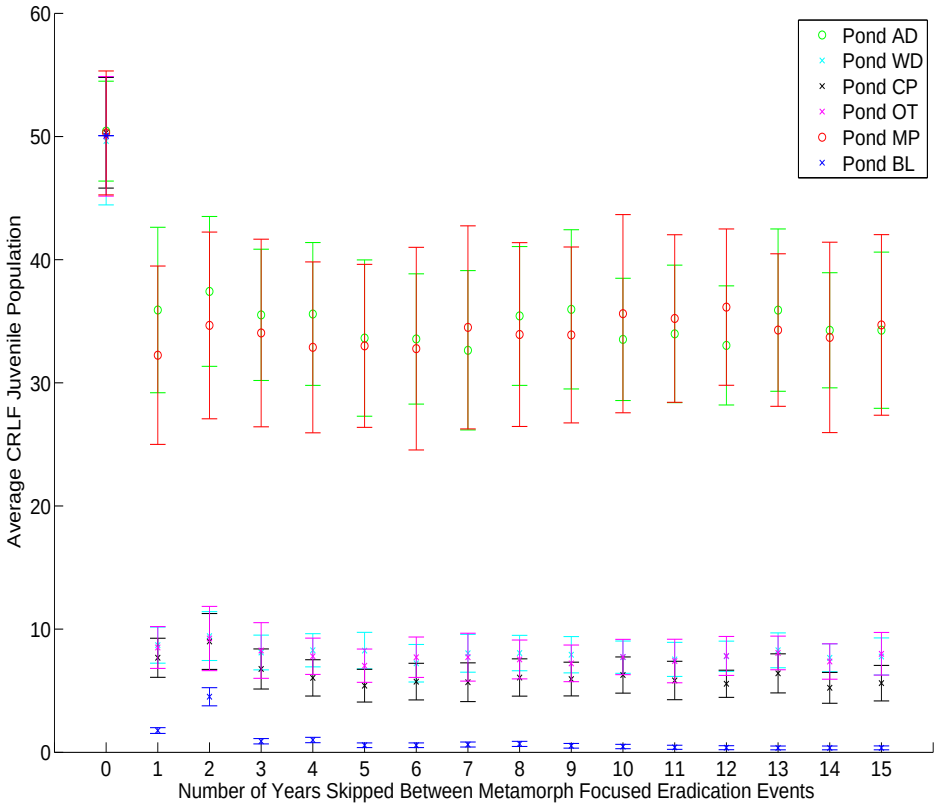


Figure 20: Skipping years between bullfrog metamorph focused eradication efforts. Skipping any years between management events has the same effect as not managing the ponds at all. Note that for all remaining figures, the seasonal ponds' averages are denoted by an 'o' while the permanent ponds' averages are marked with an 'x'.

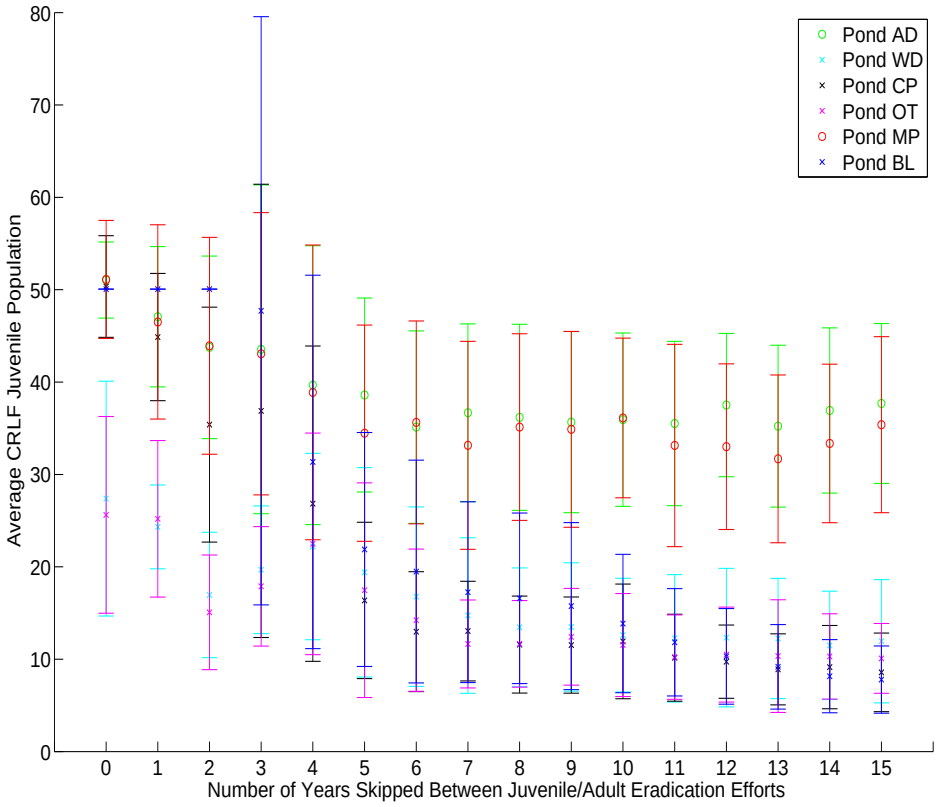


Figure 21: Skipping years between terrestrial bullfrog focused eradication efforts. As expected based on the results shown in Figure 18, the average juvenile CRLF populations at ponds WD and OT cannot be managed properly using this strategy. However, ponds CP and BL can have up to three years skipped between eradication efforts, so long as 100% of terrestrial bullfrogs can be eliminated. Note that for all remaining figures, the seasonal ponds' averages are denoted by an 'o' while the permanent ponds' averages are marked with an 'x'.

DISCUSSION

Deterministic and Stochastic Versions of the Model

Both deterministic and stochastic model versions incorporated more realistic assumptions about the life cycles of the two species than previous work by Doubledee et al. (2003). We modeled dispersal based on telemetry data of CRLFs (Fellers and Kleeman, 2007). The movement rules are distance-dependent for both the stochastic and the deterministic model. The proportion of the population that can travel a certain distance is calculated in a density-dependent way in the stochastic model. In the deterministic case, the gamma distribution is fixed and the population size has no effect on the number of animals which disperse specific distances. Furthermore, movement rules among both versions included species-specific habitat quality effects and stage-based differentiation of pond choice by bullfrog juveniles when several ponds are within dispersal range.

The deterministic version of our model produced intuitive results and showed that seasonal ponds act as a refuge for CRLFs. As expected, in the isolated seasonal ponds (removed from figures to simplify results) bullfrogs were not present and in isolated permanent ponds, CRLFs died out at the predetermined time (in our case, at year 60). Through dispersal, connected seasonal ponds act as a refuge allowing longer coexistence than would be possible in one pond. The refuge effect has been investigated before and it is known to be an important factor in facilitating the survival of rare and threatened species (summarized by Edelstein-Keshet, 1988).

The stochastic model contains all of the features of the deterministic model with added variablity. The incorporation of density dependence in the stochastic model's movement rule enabled us to see extinction and re-colonization events (e.g., see pond CP's trajectory around year 80 in Figure 15). The structure of the stochastic version allows for a greater

likelihood of longer dispersal events when populations are larger. We see higher average CRLF population sizes for all stages in all study ponds in the stochastic model than the deterministic model. The importance of incorporating these factors led us to continue using the stochastic model when implementing our management strategies.

In their paper, Doubledee et al. (2003) derive an analytical expression that shows when coexistence is guaranteed. The expression sets a bound on the sum of the bullfrog attack rates on CRLFs in terms of the other model parameters, indicating that coexistence occurs between these species when bullfrog predation is limited. We were able to find a connection between coexistence and our attack rates (α triplet) numerically. According to the choice of α triplet, specific durations of coexistence within one pond can be chosen. Furthermore, with the expansion of our one pond model to several ponds, and allowing dispersal between, we found that coexistence can be achieved in connected ponds despite the fact that the α triplet predicted a clear end to the CRLF populations in a single pond.

Tadpole Focused Eradication

For the seasonal ponds, removing bullfrog tadpoles has little effect on the average juvenile CRLF population and, given that funds for such management may be low, may be skipped altogether. We attribute this phenomenon to the fact that the bullfrog tadpoles are already being eradicated via the drying of these ponds every year. However, in the permanent ponds, $\approx 75\%$ eradication of bullfrog tadpoles is needed to bring CRLF juvenile populations up to the standard set by the seasonal ponds when no management is implemented.

Implementing the management every year obviously gives the best result; every pond's average CRLF population is at its optimum (Figure 19). Skipping two years between eradication efforts is the next best choice according to the averages presented. However, the

variance for the two year skip scheme is quite large. Skipping one year has less variance but the averages are lower. One might argue that skipping one or two years has qualitatively similar results. Although the two year skip scenario has the larger average, one will more often encounter 'bad years' (when compared to the one year skip scheme) since the variance is so high. Always, our goal is to try to lift the permanent pond's populations up to as close to the level of the seasonal pond's population sizes as possible. Skipping one or two years for this strategy appears to be reasonable, although skipping two years is clearly more risky. Skipping three years in a row enacts a clear divergence in the average CRLF population sizes between seasonal and permanent ponds, and thus we do not recommend skipping more than two years in a row.

Metamorph Focused Eradication

For the metamorph focused eradication scenario as well, wildlife technicians could get away with not managing the seasonal ponds, as they act as a refuge for the CRLF. However, in order to match the unmanaged seasonal ponds in average juvenile population size, the permanent ponds would have to have at least 90% of their metamorphic bullfrogs removed as they emerge from ponds after metamorphosis. Clearly, no years can be skipped if this is the chosen avenue of management (see figure 20). We feel that due to the high volume of eradication needed, the fact that bullfrog tadpoles do not often finish metamorphosis at the same time (Collins, 1979), and that implementation must happen every year to be worthwhile, we find this strategy to be relatively ineffective.

Juvenile/Adult Focused Eradication

With this strategy, different ponds require different levels of management. The two seasonal ponds require nothing in the way of eradication, as we have seen with the other strategies. However, there seem to be two different groups of permanent ponds. Ponds CT and BL are managed suitably by eradicating at least 50% of terrestrial frogs, while ponds WD and OT cannot, even if 100% of the terrestrial bullfrogs are eradicated. We attribute this to the way events are ordered within the simulation. The eradication effort should occur while the bullfrogs are at their breeding areas ensuring the highest volume of removal. The frogs in the simulation are breeding before they are being eradicated, which is certainly a reasonable assumption when considering the reality of implementing this strategy. This phenomenon allows for the perpetuation of future generations even if all breeding adults are eventually taken. Furthermore, ponds CT and BL share a geographical feature that aids in their manageability. They are both connected to seasonal ponds. First year juvenile bullfrogs are traveling to these seasonal ponds only to be removed before they have the ability to move to a permanent pond as a second year juvenile in order to breed as an adult.

When skipping years between implementations of this strategy, results are similar as when we do not skip any years: ponds WD and OT maintain lower averages. Skipping one year has little effect on the averages of any pond. Pond BL can have up to two years skipped in its management without changing the pond's average population (see figure 21).

Conclusions

Based on the results of our simulations, seasonal ponds do not require management since average juvenile CRLF populations were found to be sustainable. Tadpole focused eradication is effective for managing all permanent ponds so long as either at least 75% of bullfrog

tadpoles are removed every year or that all bullfrog tadpoles are removed (via draining the pond, perhaps) at least every two years. Some ponds (CT and BL) were found to be managed sufficiently via terrestrial frog removal. However, it may be cumbersome to determine the size of the terrestrial frog population and, perhaps more importantly, detect if at least 50% of them have been removed.

We are aware of two works (Govindarajulu et al., 2005; Doubledee et al., 2003) which deal specifically with contolling invasive bullfrogs. Govindarajulu et al. (2005) suggest that the best way to control bullfrog populations is to cull metamorphic bullfrogs every year, which we found to be the least effective approach. On the other hand, Doubledee et al. (2003) contend that removing tadpoles in conjunction with taking adult bullfrogs via shooting is the best strategy for invasive control. We did not test the combination of these strategies (addressed in Future Work), and therefore cannot make any comparison at this time. We were able to conclude that removing at least 50% of terrestrial bullfrogs in ponds CT and BL was an effective means of control for those permanent ponds only. However, since the strategy did not work for all permanent ponds, we cannot recommend this strategy overall. Practicality of this strategy may also pose a problem to managers. Doubledee et al. (2003) acknowledge that removing adults requires an exorbitant amount of effort and therefore may not be a feasible strategy on its own. Since terrestrial frog removal would require a possibly extensive search effort, we find it to be impractical relative to the tadpole eradication strategy, which is bound spatially by ponds.

Based on our results and the reality of the problem we offer two avenues of bullfrog management: 1) annually removing at least 75% of bullfrog tadpoles from permanent ponds, or 2) draining permanent ponds (removing 100% of bullfrog tadpoles) at least every two years. So long as recommendation 2) is done after CRLF tadpoles have all finished metamorphosis, one can be sure that all bullfrog tadpoles are removed and that no CRLFs

are harmed. Furthermore, an added bonus would be that any predatory fish would be removed as well.

Future Work

Over the course of this project, we discovered several ways in which the model could be refined. We found that habitat quality can be described by much more than just the hydroperiod of the pond. Characteristics such as pond depth, average water temperature during the breeding season, and amount of emergent vegetation are excellent indicators of the quality of the habitat and would play a role in the choice of a future breeding pond made by second year juvenile bullfrogs. Furthermore, the hydroperiod of ponds is not often as cut and dry as 'seasonal' or 'permanent'. Often times, ponds will have hydroperiods that exhibit both: drying up two years in a row and not drying up the third year, for example. A more dynamic hydroperiod index could be incorporated to reflect this phenomenon.

We chose to implement our movement rule under the notion that juvenile frogs are the primary dispersers among all life stages (Lannoo, 2005), and therefore enabled only those individuals to disperse. We translated the notion of high philopatry exhibited by these species to mean that adults will return to their breeding pond every year, although they may migrate to other ponds throughout the year to meet their needs (Semlitsch, 2008). Since we were only concerned with where frogs are located during the census (which occurs during the beginning of the birth pulse) adults movements outside the census are inconsequential. The data (Fellers and Kleeman, 2007) were collected using telemetry and thus only included adult frogs due to the heft of the transmitter. With advancements in technology, perhaps future projects will be able to track the dispersal of juvenile frogs and thus more appropriate data could be incorporated.

The design of certain elements of our movement rules (i.e., juvenile stage-specific choice probability and species-specific habitat quality indicators) was created in an entirely theoretical realm. Not having hard probabilities for such phenomena led us to theoretical estimates to describe these events. With the continuation of field research on these species, these parameters may be uncovered. In that event, our model could be greatly strengthened by their inclusion.

In our model, we simplified the survival and interaction ranges put forth by Doubledee et al. (2003) and Govindarajulu et al. (2005) by using only the published values (see Table 1), which are the averages among the ranges. Stochasticity could be implemented into these rates which would reflect the variability implied by the range of possible values.

We investigated our management strategies by implementing one stage-focused eradication effort at a time over all ponds. Combining the strategies every year or switching between strategies from year to year may be more effective than simply sticking to one method season after season. Depending on the effectiveness of certain combinations, perhaps more years can be skipped between management events.

A specific α triplet was chosen to control the coexistence duration for our management simulations. Not having this information, we chose a duration (60 years) that seemed appropriate. When it is known how long these two species can coexist in a shared habitat, a perhaps more appropriate α triplet can be chosen which may or may not change the management decision. To conclude, even if these results are not exactly accurate due to our choice of the α triplet, this project has nonetheless exercised the method of appropriate model design. We have learned from this work that ecologically significant information can be collected in order to construct a model which produces results that enable us to decide the best course of action toward sustaining a threatened amphibian species despite the invasion of an adaptable competitor/predator based on the best available science.

BIBLIOGRAPHY

Adams, M. J. 1999. Correlated Factors in Amphibian Decline: Exotic Species and Habitat Change in Western Washington. *The Journal of Wildlife Management*, **63**(4).

Alford, R. A. and S. J. Richards. 1999. Global Amphibian Declines: A Problem in Applied Ecology. *Annual Review of Ecology and Systematics*, **30**.

Blaustein, A. R. and B. A. Bancroft. 2007. Amphibian Population Declines: Evolutionary Considerations. *BioScience*, **57**(5).

Blaustein, A. R. and J. M. Kiesecker. 1998. Effects of Introduced Bullfrogs and Smallmouth Bass on Microhabitat Use, Growth, and Survival of Native Red-legged Frogs (*Rana aurora*). *Conservation Biology*, **12**(4).

Blaustein, A. R., J. M. Romansic, J. M. Kiesecker, and A. C. Hatch. 2003. Ultraviolet Radiation, Toxic Chemicals and Amphibian Population Declines. *Diversity and Distributions*, **9**(2).

Bury, B. R. and J. Whelan. 1984. Ecology and Management of the Bullfrog. *U.S. Fish and Wildlife Service Resource Publication*, **155**.

Cecil, S. G. and J. J. Just. 1979. Survival Rate, Population Density and Development of Naturally Occurring Anuran Larvae (*Rana catesbeiana*). *Copeia*, **1979**(3).

Chatwin, T. and P. Govindarajulu. 2006. Survey of Bullfrogs *Rana catesbeiana* in British Columbia. *University of Victoria*, **http://web.uvic.ca/bullfrogs/**.

Collins, J. P. 1979. Intrapopulation Variation in the Body Size at Metamorphosis and Timing of Metamorphosis in the Bullfrog, *Rana catesbeiana*. *Ecology*, **60**(4).

Davidson, C., H. B. Shaffer, and M. R. Jennings. 2001. Declines of the California Red-legged Frog: Climate, UV-B, Habitat, and Pesticides Hypotheses. *Ecological Applications*, **11**(2).

Doubledee, R. A., E. B. Muller, and R. M. Nisbet. 2003. Bullfrogs, Disturbance Regimes, and the Persistence of California Red-legged Frogs. *Journal of Wildlife Management*, **67**(2).

Edelstein-Keshet, L. 1988. *Mathematical models in biology*. New York : Random House.

Fellers, G. M. and P. M. Kleeman. 2007. California Red-legged Frog (*Rana draytonii*) Movement and Habitat Use: Implications for Conservation. *Journal of Herpetology*, **41**(2).

Franklin, A. B., B. R. Noon, and T. L. George. 2002. What is Habitat Fragmentation? *Studies in Avian Biology*, **25**.

Govindarajulu, P., R. Altwegg, and B. R. Anholt. 2005. Matrix Model Investigation of Invasive Species Control: Bullfrogs on Vancouver Island. *Ecological Applications*.

Hayes, M. P. and M. R. Jennings. 1986. Decline of Ranid Frog Species in Western North America: Are Bullfrogs (*Rana catesbeiana*) Responsible? *Journal of Herpetology*, **20**(4).

Herbold, B. and P. B. Moyle. 1986. Introduced Species and Vacant Niches. *The American Naturalist*, **128**(5).

Jennings, M. R. and M. P. Hayes. 1985. Pre-1900 Overharvest of California Red-Legged Frogs (*Rana aurora draytonii*): The Inducement for Bullfrog (*Rana catesbeiana*) Introduction. *Herpetologica*, **41**(1).

Jennings, M. R. and M. P. Hayes. 1994. Amphibian and reptile species of special concern in California. *California Department of Fish and Game, Inland Fisheries Division, Rancho Cordova, California, USA.*

Kats, L. B. and R. P. Ferrer. 2003. Alien Predators and Amphibian Declines: Review of Two Decades of Science and the Transition to Conservation. *Diversity and Distributions*, **9**(2).

Kiesecker, J. and A. R. Blaustein. 1997. Population Differences in Responces of Red-legged Frogs (*Rana aurora*) to Introduced Bullfrogs. *Ecology*, **78**.

Kupferberg, S. J. 1997. Bullfrog (*Rana catesbeiana*) Invasion of a California River: The Role of Larval Competition. *Ecology*, **78**.

Lannoo, M. 2005. *Amphibian Declines: The Conservation Status of United States Species*. University of California Press, Berkeley and Los Angeles, California.

Licht, E. L. 1974. Survival of embryos, tadpoles, and adults of the frogs *Rana aurora* and *Rana pretiosa pretiosa* sympatric in southwestern British Columbia. *Canadian Journal of Zoology*, **52**(5).

Manchester, S. J. and J. M. Bullock. 2000. The impacts of non-native species on UK biodiversity and the effectiveness of control. *Journal of Applied Ecology*, **37**(5).

Marsh, D. M. and P. C. Trenham. 2001. Metapopulation Dynamics and Amphibian Conservation. *Conservation Biology*, **15**(1).

Moyle, P. B. 1973. Effects of Introduced Bullfrogs, *Rana catesbeiana*, on the Native Frogs of San Joaquin Valley, California. *Copeia*, **1973**(1).

Raney, E. C. 1940. Summer Movements of the Bullfrog, *Rana catesbeiana* Shaw, as Determined by the Jaw-Tag Method. *American Midland Naturalist*, **23**(3).

Rice, J. A. 1995. *Mathematical Statistics and Data Analysis*. Duxbury Press.

Semlitsch, R. D. 2008. Differentiating Migration and Dispersal Processes for Pond-Breeding Amphibians. *The Journal of Wildlife Management*, **72**(1).

Stinner, J., N. Zarlinga, and S. Orcutt. 1994. Overwintering Behavior of Adult Bullfrogs, *Rana catesbeiana*, in Northeastern Ohio. *Ohio Journal of Science*, **94**(1).

Zonick, C. 2005. A Monitoring Program to Track the Effects of Habitat Restoration on Pond-breeding Amphibian Populations at Metro Natural Areas. *Metro Regional Parks and Greenspaces*.

APPENDIX

Here we provide the reader with our simulation codes. Each individual code begins and ends with an entire line of %s.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code creates possible alpha triplets based on what coexistance
% scheme we are shooting for.  Call AlphaTripletsFigure.m afterward to
% create the figure for it.

% Number of ponds.
nponds = 1;

% Initial conditions.
u = [4956 42 4];
v = [9158 754 68 16 4];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u;
catesbeiana(:,1) = v;

% Turning the row vector into a column vector.
draytonii(:,1) = draytonii(:,1)';
%catesbeiana(:,1) = catesbeiana(:,1)';

% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;
```

```
% Parameters
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% The alphaVector holds all the possibilities of values of alphaM0,M1,M2
% that we are willing to test.
alphaVector = [0.05 0.04 0.03 0.02 0.01 0.009 0.008 0.007 0.006 0.005
0.004 0.003 0.002 0.001 0.0009 0.0008 0.0007 0.0006 0.0005 0.0004 0.0003
0.0002 0.0001 0.00009 0.00008 0.00007 0.00006 0.00005 0.00004 0.00003
0.00002 0.00001];

for p = 1:32
    for q = 1:32
        for o = 1:32
            alphaM0 = alphaVector(p);
            alphaM1 = alphaVector(q);
            alphaM2 = alphaVector(o);

            for i = 1:nsteps
                for j = 1:nponds

        % Assigning the constant entries.
                    D(j*3-2,3*j) = r*p4;
                    D(3*j,3*j) = p4;
                    C(j*5-1,j*5-2) = s3;
                    C(j*5,j*5) = s5;
                    C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
                    C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
                end

    % Assigning the function entries.

                for j = 1:nponds
                    D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1*
catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
                    D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
```

```
                    C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
                    C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
                    C(j*5,j*5-1) = s4;
                end


                % Here we step from one year to the next.
                    draytonii(:,i+1) = D*draytonii(:,i);
                    catesbeiana(:,i+1) = C*catesbeiana(:,i);

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.

                    neg_d=find(draytonii<0); draytonii(neg_d)=0;
                    neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;


            end

   % Here's where we collect our data for each simulation run.
   if draytonii>1
   fid = fopen(['alphaM0_' num2str(nsteps) '.dat' ],'a');
%fprintf(fid, '%3.6f ', i)
   fprintf(fid, '%3.6f ', alphaM0);fprintf(fid,'\n');
   fclose(fid);
   fid = fopen(['alphaM1_' num2str(nsteps) '.dat' ],'a');
%fprintf(fid, '%3.6f ', i);
   fprintf(fid, '%3.6f ', alphaM1);fprintf(fid,'\n');
   fclose(fid);
   fid = fopen(['alphaM2_' num2str(nsteps) '.dat' ],'a');
%fprintf(fid, '%3.6f ', i);
   fprintf(fid, '%3.6f ', alphaM2);fprintf(fid,'\n');
   fclose(fid);
   end
```

```
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code makes the figure for AlphaTriplets.m

% Calling files.
alphaM0_20Vector = load(['alphaM0_20.dat']);
alphaM1_20Vector = load(['alphaM1_20.dat']);
alphaM2_20Vector = load(['alphaM2_20.dat']);

alphaM0_40Vector = load(['alphaM0_40.dat']);
alphaM1_40Vector = load(['alphaM1_40.dat']);
alphaM2_40Vector = load(['alphaM2_40.dat']);

alphaM0_60Vector = load(['alphaM0_60.dat']);
alphaM1_60Vector = load(['alphaM1_60.dat']);
alphaM2_60Vector = load(['alphaM2_60.dat']);

alphaM0_80Vector = load(['alphaM0_80.dat']);
alphaM1_80Vector = load(['alphaM1_80.dat']);
alphaM2_80Vector = load(['alphaM2_80.dat']);

alphaM0_100Vector = load(['alphaM0_100.dat']);
alphaM1_100Vector = load(['alphaM1_100.dat']);
alphaM2_100Vector = load(['alphaM2_100.dat']);

alphaM0_200Vector = load(['alphaM0_200.dat']);
alphaM1_200Vector = load(['alphaM1_200.dat']);
alphaM2_200Vector = load(['alphaM2_200.dat']);

% And here's our figure.
figure(1)
grid on
hold on
```

```
plot3(alphaM0_20Vector, alphaM1_20Vector, alphaM2_20Vector,'k.');%,'b.');
plot3(alphaM0_40Vector, alphaM1_40Vector, alphaM2_40Vector,'b.');%,'go');
plot3(alphaM0_60Vector, alphaM1_60Vector, alphaM2_60Vector,'r.');%,'rx');
plot3(alphaM0_80Vector, alphaM1_80Vector, alphaM2_80Vector,'g.');%,'cs');
plot3(alphaM0_100Vector, alphaM1_100Vector, alphaM2_100Vector,'m.');%,'m^');
plot3(alphaM0_200Vector, alphaM1_200Vector, alphaM2_200Vector,'c.');%,'kp');

set(gca,'zscale','log'); set(gca,'yscale','log'); set(gca,'xscale','log');
axis([0 0.0003 0 0.06 0 0.05]);
xlabel('alphaD0'); ylabel('alphaD1'); zlabel('alphaD2');
legend('20 Year Coexistence', '40 Year Coexistence', '60 Year Coexistence',
'80 Year Coexistence', '100 Year Coexistence', '200 Year Coexistence');
title('Possible Unknown Parameter Triplets for Various Coexistence Scenarios');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code simulates interacting Lithobates catesbeianus and Rana
% draytonii in a landscape composed of eight ponds.  The model is based off
% of matrix equations presented by Doubledee et al (2003). It has been
% modified to expand the number of stages for the bullfrog, contract the
% number of stages for the California red-legged frog (CRLF), incorporate
% tadpole interactions between species and a 'fast-track' tadpole option
% for the first year tadpoles of the bullfrog.

% First, create a matrix (M) which holds all the distances between all
% possible pairs of ponds.  Next, we create matrices R_C1, R_C2, and R_D
% which hold rates of movement for first year bullfrog juveniles, second
% year bullfrog juveniles, and CRLF juveniles, respectively.  They were
% calculated using a gamma distribution fitted to the data of Fellers and
% Kleeman (2007).  These probabilities were also multiplied by decision
% probabilities differentiated by stage, and habitat quality movement
% probabilities that are unique over species. This allows proportions of
% the frog populations to move according to what has been observed in the
% field according to the best available science.  This code finishes with
% two time series figures, one with three subfigures encompassing the three
% CRLF stages, and one with five subfigures encompassing the five bullfrog
```

```
% stages.
%-------------------------------------------------------------------------

% Number of ponds.
nponds = 8;

% Pond distance matrix initialization.
M = zeros(nponds,nponds);

% Pond distance matrix entry values.
M(1,2) =  14740;
M(2,1) = M(1,2);
M(1,3) =  15240;
M(3,1) = M(1,3);
M(1,4) =  15640;
M(4,1) = M(1,4);
M(1,5) =  21610;
M(5,1) = M(1,5);
M(1,6) =  22400;
M(6,1) = M(1,6);
M(1,7) =  30830;
M(7,1) = M(1,7);
M(1,8) =  43870;
M(8,1) = M(1,8);

M(2,3) =  519.38;
M(3,2) = M(2,3);
M(2,4) =  885.14;
M(4,2) = M(2,4);
M(2,5) =  6950;
M(5,2) = M(2,5);
M(2,6) =  7720;
M(6,2) = M(2,6);
M(2,7) =  16130;
M(7,2) = M(2,7);
M(2,8) =  29130;
M(8,2) = M(2,8);
```

```
M(3,4) =  541.12;
M(4,3) = M(3,4);
M(3,5) =  6440;
M(5,3) = M(3,5);
M(3,6) =  7210;
M(6,3) = M(3,6);
M(3,7) =  15610;
M(7,3) = M(3,7);
M(3,8) =  28650;
M(8,3) = M(3,8);

M(4,5) =  6040;
M(5,4) = M(4,5);
M(4,6) =  6840;
M(6,4) = M(4,6);
M(4,7) =  15210;
M(7,4) = M(4,7);
M(4,8) =  28210;
M(8,4) = M(4,8);

M(5,6) =  677.97;
M(6,5) = M(5,6);
M(5,7) =  9110;
M(7,5) = M(5,7);
M(5,8) =  22270;
M(8,5) = M(5,8);

M(6,7) =  8400;
M(7,6) = M(6,7);
M(6,8) =  21580;
M(8,6) = M(6,8);

M(7,8) =  13340;
M(8,7) = M(7,8);

MD = M;
```

```
MC = M;

% This forces the distance entries to be less that 5600 meters, our assumed
% maximum traversable distance for the bullfrog.
for i = 1:nponds
    for j = 1:nponds
        if MC(i,j) > 5600
            MC(i,j) = 0;
        end
    end
end

% This forces the distance entries to be less that 2800 meters, our assumed
% maximum traversable distance for the CRLF.
for i = 1:nponds
    for j = 1:nponds
        if MD(i,j) > 2800
            MD(i,j) = 0;
        end
    end
end

% Here we create our gamma distributions for both species.  Note that since
% we are in the 'deterministic' version of the simulation, we only create
% one gamma distribution for each species. The distributions  are dependent
% only on the distances between the ponds.

[R_D]=calculate_rij_det_Dnew(MD);
[R_C1]=calculate_rij_det_Cnew(MC);
[R_C2]=calculate_rij_det_Cnew(MC);

newMC2 = zeros(nponds,nponds);
RecipnewMC2 = zeros(nponds,nponds);
    [rowC2,colC2] = find(R_C2);
    DispDepInflC2 = zeros(nponds,nponds);
    for g = 1:length(rowC2)
            newMC2(rowC2(g),colC2(g)) = MC(rowC2(g),colC2(g));
```

```
end
for h = 1:nponds

    if nnz(R_D(h,:)) > 1
        R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
    end
    if nnz(R_C1(h,:)) > 1
        R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
    end
    for k = 1:nponds
        if newMC2(h,k)==0
            DispDepInflC2(h,k) = 0;

        else
            RecipnewMC2(h,k) = 1/newMC2(h,k);

        end
    end
end

for h = 1:nponds
    for k = 1:nponds
        if RecipnewMC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewMC2(h,k))/(sum(RecipnewMC2(h,:)));
        end
    end
end
 for j = 1:nponds

     for k = 1:nponds
         R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
     end
 end

for j = 1:length(rowC2)
         if colC2(j)==1
            R_C2(rowC2(j),colC2(j))=0;
```

```
            elseif colC2(j)==2
               R_C2(rowC2(j),colC2(j))=0;
            elseif colC2(j)==6
               R_C2(rowC2(j),colC2(j))=0;
            elseif colC2(j)==7
               R_C2(rowC2(j),colC2(j))=0;
            end
    end

     R_Dtrans = R_D';
     R_C1trans = R_C1';
     R_C2trans = R_C2';

% Initial conditions.

u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
4956 42 4 ];
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.

draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.

draytonii(:,1) = u;
catesbeiana(:,1) = v;

% Turning the row vector into a column vector.

draytonii(:,1) = draytonii(:,1)';
catesbeiana(:,1) = catesbeiana(:,1)';

% Creating zero matrices.
```

```
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.

nsteps = 200;


% Parameters

p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; sFT = 0.016; s2 = 0.02; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

for i = 1:nsteps
    for j = 1:nponds

        % Assigning the constant entries.

        D(j*3-2,3*j) = r*p4;
        D(3*j,3*j) = p4;
        C(j*5-1,j*5-2) = s3;
        C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.

        if j == 1
            C(j*5-3,j*5-4) = 0;
        elseif j == 2
```

```
            C(j*5-3,j*5-4) = 0;
        elseif j == 6
            C(j*5-3,j*5-4) = 0;
        elseif j == 7
            C(j*5-3,j*5-4) = 0;
        else
            C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
        end


        C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
    end

    % Assigning the function entries.

    for j = 1:nponds


            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5,j*5-1) = s4;



        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.

        Juv_D(j,1)=draytonii(j*3-1,i);

        Juv_C1(j,1)=catesbeiana(j*5-2,i);
        Juv_C2(j,1)=catesbeiana(j*5-1,i);

        % Here we gather the adult population of bullfrogs in order to
        % calculate the CRLF habitat quality movement probability.
```

```
    Ad_C(j,1) = catesbeiana(j*5,i);
    if Ad_C(j)==0
        Ad_C(j)=1;
    end
    if Ad_C(j)<1
        Ad_C(j)=1;
    else
        Ad_C(j)=1-(1/(Ad_C(j)));
    end
     if Ad_C(j)==0
        Ad_C(j)=1;
    end
end

%Here we incorporate this probability into our movement rate matrix.
for j = i:nponds
     for k = 1:nponds
         if Ad_C(j)~=0
             R_D(j,k) = R_D(j,k)*Ad_C(j);
         end
     end
 end
% Multiplying our juvenile population by the proportion of moving
% indivuals gives us the number of individuals leaving their pond.

for h = 1:nponds
    immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
    immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
    immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);

end

immVectorJuvD = sum(immMatrixJuvD,2);
immVectorJuvC1 = sum(immMatrixJuvC1,2);
immVectorJuvC2 = sum(immMatrixJuvC2,2);
emmVectorJuvD = R_Dtrans*Juv_D;
```

```
    emmVectorJuvC1 = R_C1trans*Juv_C1;
    emmVectorJuvC2 = R_C2trans*Juv_C2;

    % The following three paragraphs are present mearly to put all the
    % above immigration/emigration information in vectors of the
    % appropriate size to be subtracted from and added to the population
    % vector.

    x_JuvD_dummy=[0 0 1];
    imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
    emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

    x_JuvC1_dummy=[0 0 0 1 0];
    imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
    emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

    x_JuvC2_dummy=[0 0 0 0 1];
    imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
    emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
    % the immigration/emigration rules outlined above.

    draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
    catesbeiana(:,i+1) = C*catesbeiana(:,i)-imm_JuvC1'+emm_JuvC1'-imm_JuvC2'
    +emm_JuvC2';

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
    neg_d=find(draytonii<0); draytonii(neg_d)=0;
    neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;
end

% Here's the way we create the figures.

t=(0:1:nsteps);
```

```
figure(1)
subplot(3,1,1)
hold on
%plot(t,draytonii(1,:),'m');
plot(t,draytonii(4,:),'g');
plot(t,draytonii(7,:),'c');
plot(t,draytonii(10,:),'k');
plot(t,draytonii(13,:),'m');
plot(t,draytonii(16,:),'r');
%plot(t,draytonii(19,:),'m');
plot(t,draytonii(22,:),'b');
title('CRLF Tadpole Population');
xlabel('Years');
ylabel('Number of Individuals');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');


subplot(3,1,2)
hold on
%plot(t,draytonii(2,:),'m');
plot(t,draytonii(5,:),'g');
plot(t,draytonii(8,:),'c');
plot(t,draytonii(11,:),'k');
plot(t,draytonii(14,:),'m');
plot(t,draytonii(17,:),'r');
%plot(t,draytonii(20,:),'m');
plot(t,draytonii(23,:),'b');
title('CRLF Juvenile Population');
xlabel('Years');
ylabel('Number of Individuals');
%legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');


subplot(3,1,3)
hold on
%plot(t,draytonii(2,:),'m');
plot(t,draytonii(6,:),'g');
```

```
plot(t,draytonii(9,:),'c');
plot(t,draytonii(12,:),'k');
plot(t,draytonii(15,:),'m');
plot(t,draytonii(18,:),'r');
%plot(t,draytonii(21,:),'m');
plot(t,draytonii(24,:),'b');
title('CRLF Adult Population');
xlabel('Years');
ylabel('Number of Individuals');
%legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');

figure(2)
subplot(5,1,1)
hold on
plot(t,catesbeiana(6,:),'g');
plot(t,catesbeiana(11,:),'c');
plot(t,catesbeiana(16,:),'k');
plot(t,catesbeiana(21,:),'m');
plot(t,catesbeiana(26,:),'r');
%plot(t,draytonii(21,:),'m');
plot(t,catesbeiana(36,:),'b');
title('BF First Year Tadpole Population');
xlabel('Years');
ylabel('# of Individuals');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');

subplot(5,1,2)
hold on
plot(t,catesbeiana(7,:),'g');
plot(t,catesbeiana(12,:),'c');
plot(t,catesbeiana(17,:),'k');
plot(t,catesbeiana(22,:),'m');
plot(t,catesbeiana(27,:),'r');
%plot(t,draytonii(21,:),'m');
plot(t,catesbeiana(37,:),'b');
title('BF Second Year Tadpole Population');
xlabel('Years');
```

```
ylabel('# of Individuals');
%legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');

subplot(5,1,3)
hold on
plot(t,catesbeiana(8,:),'g');
plot(t,catesbeiana(13,:),'c');
plot(t,catesbeiana(18,:),'k');
plot(t,catesbeiana(23,:),'m');
plot(t,catesbeiana(28,:),'r');
%plot(t,draytonii(21,:),'m');
plot(t,catesbeiana(38,:),'b');
title('BF First Year Juvenile Population');
xlabel('Years');
ylabel('# of Individuals');
%legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');

subplot(5,1,4)
hold on
plot(t,catesbeiana(9,:),'g');
plot(t,catesbeiana(14,:),'c');
plot(t,catesbeiana(19,:),'k');
plot(t,catesbeiana(24,:),'m');
plot(t,catesbeiana(29,:),'r');
%plot(t,draytonii(21,:),'m');
plot(t,catesbeiana(39,:),'b');
title('BF Second Year Juvenile Population');
xlabel('Years');
ylabel('# of Individuals');
%legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');

subplot(5,1,5)
hold on
plot(t,catesbeiana(10,:),'g');
plot(t,catesbeiana(15,:),'c');
plot(t,catesbeiana(20,:),'k');
plot(t,catesbeiana(25,:),'m');
```

```
plot(t,catesbeiana(30,:),’r’);
%plot(t,draytonii(21,:),’m’);
plot(t,catesbeiana(40,:),’b’);
title(’BF Adult Population’);
xlabel(’Years’);
ylabel(’# of Individuals’);
%legend(’Pond AD’, ’Pond WD’, ’Pond CP’, ’Pond OT’, ’Pond MP’, ’Pond BL’);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [R]=calculate_rij_det_Cnew(Dnbyn)

% This code calculates our Fellers and Kleeman (2007) data fitted
% gamma distribution for an element of our bullfrog movment rule
% in the deterministic model.

nponds = 8;

R = zeros(nponds,nponds);

for i = 1:nponds
     for j = 1:nponds


         if Dnbyn(i,j) == 0
            R(i,j)=0;
         else
            D=Dnbyn(i,j);
            p = gamcdf(D,0.3911,800);
            R(i,j) = 1-p;
         end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [R]=calculate_rij_det_Dnew(Dnbyn)
```

```
% This code calculates our Fellers and Kleeman (2007) data fitted
% gamma distribution for an element of our CRLF movment rule
% in the deterministic model.

nponds = 8;

R = zeros(nponds,nponds);

for i = 1:nponds
     for j = 1:nponds

          if Dnbyn(i,j) == 0
             R(i,j)=0;
          else
             D=Dnbyn(i,j);
             p = gamcdf(D,0.3911,381.6206);
             R(i,j) = 1-p;
          end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p,plo,pup] = gamcdf(x,a,b,pcov,alpha)
%GAMCDF Gamma cumulative distribution function.
%   P = GAMCDF(X,A,B) returns the gamma cumulative distribution function
%   with shape and scale parameters A and B, respectively, at the values in
%   X.  The size of P is the common size of the input arguments.  A scalar
%   input functions as a constant matrix of the same size as the other
%   inputs.
%
%   Some references refer to the gamma distribution with a single
%   parameter.  This corresponds to the default of B = 1.
%
%   [P,PLO,PUP] = GAMCDF(X,A,B,PCOV,ALPHA) produces confidence bounds for
%   P when the input parameters A and B are estimates.  PCOV is a 2-by-2
```

```
%   matrix containing the covariance matrix of the estimated parameters.
%   ALPHA has a default value of 0.05, and specifies 100*(1-ALPHA)%
%   confidence bounds.  PLO and PUP are arrays of the same size as P
%   containing the lower and upper confidence bounds.
%
%   See also GAMFIT, GAMINV, GAMLIKE, GAMPDF, GAMRND, GAMSTAT, GAMMAINC.

%   GAMMAINC does computational work.

%   References:
%      [1] Abramowitz, M. and Stegun, I.A. (1964) Handbook of Mathematical
%          Functions, Dover, New York, section 26.1.
%      [2] Evans, M., Hastings, N., and Peacock, B. (1993) Statistical
%          Distributions, 2nd ed., Wiley.

%   Copyright 1993-2004 The MathWorks, Inc.
%   $Revision: 2.12.2.4 $  $Date: 2004/12/24 20:46:50 $

if nargin < 2
    error('stats:gamcdf:TooFewInputs',...
          'Requires at least two input arguments.');
elseif nargin < 3
    b = 1;
end

% More checking if we need to compute confidence bounds.
if nargout > 1
    if nargin < 4
        error('stats:gamcdf:TooFewInputs',...
              'Must provide covariance matrix to compute confidence bounds.');
    end
    if ~isequal(size(pcov),[2 2])
        error('stats:gamcdf:BadCovariance',...
              'Covariance matrix must have 2 rows and columns.');
    end
    if nargin < 5
        alpha = 0.05;
```

```
    elseif ~isnumeric(alpha) || numel(alpha) ~= 1 || alpha <= 0 || alpha >= 1
        error('stats:gamcdf:BadAlpha',...
              'ALPHA must be a scalar between 0 and 1.');
    end
end

% Return NaN for out of range parameters.
a(a <= 0) = NaN;
b(b <= 0) = NaN;
x(x < 0) = 0;

try
    z = x ./ b;
    p = gammainc(z, a);
catch
    error('stats:gamcdf:InputSizeMismatch',...
          'Non-scalar arguments must match in size.');
end
p(z == Inf) = 1;

% Compute confidence bounds if requested.
if nargout >= 2
    % Approximate the variance of p on the logit scale
    logitp = log(p./(1-p));
    dp = 1 ./ (p.*(1-p)); % derivative of logit(p) w.r.t. p
    da = dgammainc(z,a) .* dp; % dlogitp/da = dp/da * dlogitp/dp
    db = -exp(a.*log(z)-z-gammaln(a)-log(b)) .* dp; % dlogitp/db = dp/db *
    dlogitp/dp
    varLogitp = pcov(1,1).*da.^2 + 2.*pcov(1,2).*da.*db + pcov(2,2).*db.^2;
    if any(varLogitp(:) < 0)
        error('stats:gamcdf:BadCovariance',...
              'PCOV must be a positive semi-definite matrix.');
    end

    % Use a normal approximation on the logit scale, then transform back to
    % the original CDF scale
    halfwidth = -norminv(alpha/2) * sqrt(varLogitp);
```

```
    explogitplo = exp(logitp - halfwidth);
    explogitpup = exp(logitp + halfwidth);
    plo = explogitplo ./ (1 + explogitplo);
    pup = explogitpup ./ (1 + explogitpup);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code simulates interacting Lithobates catesbeianus and Rana
% draytonii in a landscape composed of eight ponds.  The model is based off
% of matrix equations presented by Doubledee et al (2003). It has been
% modified to expand the number of stages for the bullfrog, contract the
% number of stages for the California red-legged frog (CRLF), incorporate
% tadpole interactions between species and a 'fast-track' tadpole option
% for the first year tadpoles of the bullfrog.

% First, create a matrix (M) which holds all the distances between all
% possible pairs of ponds.  Next, we create matrices R_C1, R_C2, and R_D
% which hold rates of movement for first year bullfrog juveniles, second
% year bullfrog juveniles, and CRLF juveniles, respectively.  They were
% calculated using a gamma distributed histogram fitted to the data of
% Fellers and Kleeman (2007).  These probabilities were also multiplied
% by decision probabilities differentiated by stage, and habitat quality
% movement probabilities that are unique over species. This allows
% proportions of the frog populations to move according to what has been
% observed in the field according to the best available science.  This code
% finishes with two time series figures, one with three subfigures
% encompassing the three CRLF stages, and one with five subfigures
% encompassing the five bullfrog stages.
%--------------------------------------------------------------------------

% Number of ponds.

nponds = 8;

% Pond distance matrix initialization.  Called M in the thesis.
```

```
Dnbyn = zeros(nponds,nponds);

% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);

Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
```

```
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;

% BF max distance.

for i = 1:nponds
    for j = 1:nponds
```

```
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
    end
end

% CRLF max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
end


% Initial conditions.
u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u;
catesbeiana(:,1) = v;

% Turning the row vector into a column vector.
draytonii(:,1) = draytonii(:,1)';
catesbeiana(:,1) = catesbeiana(:,1)';
```

```
% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;

% Parameters.

p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

for i = 1:nsteps

    for j = 1:nponds

        % Assigning the constant entries.

        D(j*3-2,3*j) = r*p4;
        D(3*j,3*j) = p4;
        C(j*5-1,j*5-2) = s3;
        C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.

        if j == 1
            C(j*5-3,j*5-4) = 0;
        elseif j == 2
            C(j*5-3,j*5-4) = 0;
```

```
        elseif j == 6
            C(j*5-3,j*5-4) = 0;
        elseif j == 7
            C(j*5-3,j*5-4) = 0;
        else
            C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
        end


        C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
    end

    % Assigning the function entries.

    for j = 1:nponds


        D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
        D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
        C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
        C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
        C(j*5,j*5-1) = s4;


        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.

        Juv_D(j,1)=draytonii(j*3-1,i);

        Juv_C1(j,1)=catesbeiana(j*5-2,i);
        Juv_C2(j,1)=catesbeiana(j*5-1,i);


        % CRLF habitat quality indicator.

        Ad_C(j,1) = catesbeiana(j*5,i);
```

```
    if Ad_C(j)==0
        Ad_C(j)=1;
    end
    if Ad_C(j)<1
        Ad_C(j)=1;
    else
        Ad_C(j)=1-(1/(Ad_C(j)));
    end
    if Ad_C(j)==0
        Ad_C(j)=1;
    end

end

% Here we create our gamma distributions which tells us what proportion
% of the juvenile population will move to which pond.  Note that since
% this is the 'stochastic' version of the simulation, we update the
% gamma distribution every year (time step) for each pond.
% Furthermore, each updated gamma distribution is dependent upon the
% source pond's juvenile population.  Distance values are drawn and
% used in their corresponding gamma distribution to obtain rates of
% dispersion (immigration out of a pond).

[R_D]=calculate_rij_stoch_Dnew(DnbynD,Juv_D);
[R_C1]=calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
[R_C2]=calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

% Stage specific choice probability caclulations.

newDnbynC2 = zeros(nponds,nponds);
RecipnewDnbynC2 = zeros(nponds,nponds);
[rowC2,colC2] = find(R_C2);
DispDepInflC2 = zeros(nponds,nponds);
for g = 1:length(rowC2)
        newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
end
```

```
for h = 1:nponds

    if nnz(R_D(h,:)) > 1
        R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
    end
    if nnz(R_C1(h,:)) > 1
        R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
    end

    for k = 1:nponds
        if newDnbynC2(h,k)==0
            DispDepInflC2(h,k) = 0;

        else
            RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);

        end
    end
end
for j = 1:length(rowC2)
         if colC2(j)==1
            R_C2(rowC2(j),colC2(j))=0;
         elseif colC2(j)==2
            R_C2(rowC2(j),colC2(j))=0;
         elseif colC2(j)==6
            R_C2(rowC2(j),colC2(j))=0;
         elseif colC2(j)==7
            R_C2(rowC2(j),colC2(j))=0;
         end

 end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
```

```
end
 for j = 1:nponds
     for k = 1:nponds
         R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
     end
 end
 for j = i:nponds
     for k = 1:nponds
         if Ad_C(j)~=0
             R_D(j,k) = R_D(j,k)*Ad_C(j);
         end
     end
 end

R_Dtrans = R_D';
R_C1trans = R_C1';
R_C2trans = R_C2';

% Immigration vector calculation.
for h = 1:nponds
    immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
    immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
    immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
end

immVectorJuvD = sum(immMatrixJuvD,2);
immVectorJuvC1 = sum(immMatrixJuvC1,2);
immVectorJuvC2 = sum(immMatrixJuvC2,2);

% Emigration vector calculation.
emmVectorJuvD = R_Dtrans*Juv_D;
emmVectorJuvC1 = R_C1trans*Juv_C1;
emmVectorJuvC2 = R_C2trans*Juv_C2;

% The following three paragraphs are used mearly to put all the above
% immigration/emigration information in vectors of the appropriate size
% to be subtracted from and added to the population vector.
```

```
    x_JuvD_dummy=[0 0 1];
    imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
    emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

    x_JuvC1_dummy=[0 0 0 1 0];
    imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
    emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

    x_JuvC2_dummy=[0 0 0 0 1];
    imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
    emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
    % the immigration/emigration rules outlined above.

    draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
    catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')
    +(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.

    neg_d=find(draytonii<0); draytonii(neg_d)=0;
    neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;

end

% Here's the way we create the figures.
t=(0:1:nsteps);

figure(1)
subplot(3,1,1)
hold on
plot(t,draytonii(4,:),'g');
plot(t,draytonii(7,:),'c');
plot(t,draytonii(10,:),'k');
```

```
plot(t,draytonii(13,:),'m');
plot(t,draytonii(16,:),'r');
plot(t,draytonii(22,:),'b');
title('CRLF Tadpole Population');
xlabel('Years');
ylabel('Number of Individuals');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');


subplot(3,1,2)
hold on
plot(t,draytonii(5,:),'g');
plot(t,draytonii(8,:),'c');
plot(t,draytonii(11,:),'k');
plot(t,draytonii(14,:),'m');
plot(t,draytonii(17,:),'r');
plot(t,draytonii(23,:),'b');
title('CRLF Juvenile Population');
xlabel('Years');
ylabel('Number of Individuals');

subplot(3,1,3)
hold on
plot(t,draytonii(6,:),'g');
plot(t,draytonii(9,:),'c');
plot(t,draytonii(12,:),'k');
plot(t,draytonii(15,:),'m');
plot(t,draytonii(18,:),'r');
plot(t,draytonii(24,:),'b');
title('CRLF Adult Population');
xlabel('Years');
ylabel('Number of Individuals');

figure(2)
subplot(5,1,1)
hold on
plot(t,catesbeiana(6,:),'g');
```

```
plot(t,catesbeiana(11,:),’c’);
plot(t,catesbeiana(16,:),’k’);
plot(t,catesbeiana(21,:),’m’);
plot(t,catesbeiana(26,:),’r’);
plot(t,catesbeiana(36,:),’b’);
title(’BF First Year Tadpole Population’);
xlabel(’Years’);
ylabel(’# of Individuals’);
legend(’Pond AD’, ’Pond WD’, ’Pond CP’, ’Pond OT’, ’Pond MP’, ’Pond BL’);

subplot(5,1,2)
hold on
plot(t,catesbeiana(7,:),’g’);
plot(t,catesbeiana(12,:),’c’);
plot(t,catesbeiana(17,:),’k’);
plot(t,catesbeiana(22,:),’m’);
plot(t,catesbeiana(27,:),’r’);
plot(t,catesbeiana(37,:),’b’);
title(’BF Second Year Tadpole Population’);
xlabel(’Years’);
ylabel(’# of Individuals’);

subplot(5,1,3)
hold on
plot(t,catesbeiana(8,:),’g’);
plot(t,catesbeiana(13,:),’c’);
plot(t,catesbeiana(18,:),’k’);
plot(t,catesbeiana(23,:),’m’);
plot(t,catesbeiana(28,:),’r’);
plot(t,catesbeiana(38,:),’b’);
title(’BF First Year Juvenile Population’);
xlabel(’Years’);
ylabel(’# of Individuals’);

subplot(5,1,4)
hold on
plot(t,catesbeiana(9,:),’g’);
```

```
plot(t,catesbeiana(14,:),'c');
plot(t,catesbeiana(19,:),'k');
plot(t,catesbeiana(24,:),'m');
plot(t,catesbeiana(29,:),'r');
plot(t,catesbeiana(39,:),'b');
title('BF Second Year Juvenile Population');
xlabel('Years');
ylabel('# of Individuals');

subplot(5,1,5)
hold on
plot(t,catesbeiana(10,:),'g');
plot(t,catesbeiana(15,:),'c');
plot(t,catesbeiana(20,:),'k');
plot(t,catesbeiana(25,:),'m');
plot(t,catesbeiana(30,:),'r');
plot(t,catesbeiana(40,:),'b');
title('BF Adult Population');
xlabel('Years');
ylabel('# of Individuals');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [R]=calculate_rij_stoch_Cnew(Dnbyn,n_pop)

% This code calculates our Fellers and Kleeman (2007) data fitted
% gamma distributioned histogram for an element of our BF movement
% rule in the stochastic model.

nponds = 8;

R = zeros(nponds,nponds);

n_pop=floor(n_pop);

for i = 1:nponds
 for j = 1:nponds
```

```
    if n_pop(i)<1
       R(i,:)=0;
    else
    dist1=gamm_rnd(n_pop(i),0.60,0.0013);  % Presumed max distance: 5600 m
    n_bin=ceil(max(dist1));
    [n,x]=hist(dist1,n_bin);
    prob_dist1=n/sum(n);
            if Dnbyn(i,j) > n_bin
                R(i,j)=0;
            elseif Dnbyn(i,j) == 0
                R(i,j)=0;
            else
                dist_ij=Dnbyn(i,j);
                index_v=find(x<dist_ij);
                max_index=max(index_v);
                int_prob_dist1=sum(prob_dist1(1:max_index));
                R(i,j) = 1-int_prob_dist1;
            end
    end
 end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [R]=calculate_rij_stoch_Dnew(Dnbyn,n_pop)

% This code calculates our Fellers and Kleeman (2007) data fitted
% gamma distributioned histogram for an element of our CRLF movement
% rule in the stochastic model.


nponds = 8;

R = zeros(nponds,nponds);

n_pop=floor(n_pop);
```

```
for i = 1:nponds

    if n_pop(i)<1
       R(i,:)=0;
    else

    dist1=gamm_rnd(n_pop(i),0.60,0.0028);  % Presumed max distance: 2800 m

    n_bin=ceil(max(dist1));
    [n,x]=hist(dist1,n_bin);
    prob_dist1=n/sum(n);

        for j = 1:nponds
            if Dnbyn(i,j) > n_bin
                R(i,j)=0;
            elseif Dnbyn(i,j) == 0
                R(i,j)=0;
            else
                dist_ij=Dnbyn(i,j);
                index_v=find(x<dist_ij);
                max_index=max(index_v);
                int_prob_dist1=sum(prob_dist1(1:max_index));
                R(i,j) = 1-int_prob_dist1;
            end
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gb = gamm_rnd(nrow,m,k)
% PURPOSE: a matrix of random draws from the gamma distribution
%-------------------------------------------------
% USAGE: r = gamm_rnd(nrow,m,k)
% where: nrow = the row size of the vector drawn
%        m = a parameter such that the mean of the gamma = m/k
%        k = a parameter (or vector) such that the variance of the gamma = m/(k^2)
```

```
%        note: m=r/2, k=2 equals chisq r random deviate
%-------------------------------------------------
% RETURNS:
%        r = an nrow x 1 vector of random numbers from the gamma distribution
% -------------------------------------------------
% SEE ALSO: gamm_inv, gamm_pdf, gamm_cdf
%-------------------------------------------------
% NOTE: written by: Michael Gordy, 15 Sept 1993
%                   mbgordy@athena.mit.edu
%-------------------------------------------------
% REFERENCES: Luc Devroye, Non-Uniform Random Variate Generation,
%            New York: Springer Verlag, 1986, ch 9.3-6.
if nargin ~= 3
error('Wrong # of arguments to gamm_rnd');
end;
ncol = 1;
gb=zeros(nrow,ncol);
if m<=1
  % Use RGS algorithm by Best, p. 426
  c=1/m;
  t=0.07+0.75*sqrt(1-m);
  b=1+exp(-t)*m/t;
  for i1=1:nrow
    for i2=1:ncol
       accept=0;
       while accept==0
          u=rand; w=rand; v=b*u;
          if v<=1
             x=t*(v^c);
             accept=((w<=((2-x)/(2+x))) | (w<=exp(-x)));
          else
             x=-log(c*t*(b-v));
             y=x/t;
             accept=(((w*(m+y-m*y))<=1) | (w<=(y^(m-1))));
          end
       end
       gb(i1,i2)=x;
```

```
    end
  end
else
  % Use Best's rejection algorithm XG, p. 410
  b=m-1;
  c=3*m-0.75;
  for i1=1:nrow
    for i2=1:ncol
       accept=0;
       while accept==0
          u=rand;  v=rand;
          w=u*(1-u);  y=sqrt(c/w)*(u-0.5);
          x=b+y;
          if x >= 0
             z=64*(w^3)*v*v;
             accept=(z<=(1-2*y*y/x)) | (log(z)<=(2*(b*log(x/b)-y)));
          end
       end
       gb(i1,i2)=x;
    end
  end
end
%gb = matdiv(gb,k);
gb=gb/k;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code simulates the act of dipnetting all ponds every year for
% various levels of eradication (0-100% at 5% incements) using the
% stochasic model version.  Running PondDrainingFigure.m afterward
% will produce the figure.

close all;
clear;

% Number of ponds.
```

```
nponds = 8;

% Pond distance matrix initialization.  We call this matrix M in the
% thesis.

Dnbyn = zeros(nponds,nponds);

% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);
```

```
Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;
```

```
% Bullfrog max distance limit.
for i = 1:nponds
    for j = 1:nponds
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
    end
end

% CRLF max distance limit.
for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
end


% Initial conditions.

u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.
% BF IC for initializing a single frog.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0];


% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.

draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
```

```
draytonii(:,1) = u';
catesbeiana(:,1) = v';

% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;
just_2nd_year_juv = zeros(nponds,nsteps+1);

% Parameters.
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown Parameters.
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

% w encompasses the range of eradication
for w = 0:0.05:1
    for i = 1:nsteps
        for j = 1:nponds
        % Assigning the constant entries.
            D(j*3-2,3*j) = r*p4;
            D(3*j,3*j) = p4;
            C(j*5-1,j*5-2) = s3;
            C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.
            if j == 1
                C(j*5-3,j*5-4) = 0;
            elseif j == 2
                C(j*5-3,j*5-4) = 0;
            elseif j == 6
```

```
                C(j*5-3,j*5-4) = 0;
            elseif j == 7
                C(j*5-3,j*5-4) = 0;
            else
                C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
            end
            C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
        end

    % Assigning the function entries.
        for j = 1:nponds
            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5,j*5-1) = s4;
            C(j*5-4,j*5) = (1-w)*b*s5*exp(-gamma*catesbeiana(5*j,i));
            C(j*5-3,j*5-4) = (1-w)*s1*exp(-gamma*catesbeiana(5*j,i));

        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.
            Juv_D(j,1)=draytonii(j*3-1,i);
            Juv_C1(j,1)=catesbeiana(j*5-2,i);
            Juv_C2(j,1)=catesbeiana(j*5-1,i);
        %  This bit calculates the habitat quality related movement
        %  probability.
            Ad_C(j,1) = catesbeiana(j*5,i);
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
            if Ad_C(j)<1
                Ad_C(j)=1;
            else
                Ad_C(j)=1-(1/(Ad_C(j)));
            end
            if Ad_C(j)==0
```

```
                Ad_C(j)=1;
            end
        end

    % Here we create our gamma distributions which tells us what proportion
    % of the juvenile population will move to which pond.  Note that since
    % this is the 'stochastic' version of the simulation, we update the
    % gamma distribution every year (time step) for each pond.
    % Furthermore, each updated gamma distribution is dependent upon the
    % source pond's juvenile population.  Distance values are drawn and
    % used to calculate proportions of moving individuals based on
    % information from their corresponding gamma distribution to obtain
    % rates of dispersion (immigration out of a pond).
        [R_D] = calculate_rij_stoch_Dnew(DnbynD,Juv_D);
        [R_C1] = calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
        [R_C2] = calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

% This part implements the stage specific movement rules.
        newDnbynC2 = zeros(nponds,nponds);
        RecipnewDnbynC2 = zeros(nponds,nponds);
        [rowC2,colC2] = find(R_C2);
        DispDepInflC2 = zeros(nponds,nponds);

        for g = 1:length(rowC2)
                newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
        end

        for h = 1:nponds

            if nnz(R_D(h,:)) > 1
                R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
            end
            if nnz(R_C1(h,:)) > 1
                R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
            end
            for k = 1:nponds
                if newDnbynC2(h,k)==0
```

```
            DispDepInflC2(h,k) = 0;
        else
            RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);
        end
    end
end
for j = 1:length(rowC2)
        if colC2(j)==1
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==2
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==6
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==7
            R_C2(rowC2(j),colC2(j))=0;
        end
end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
end
for j = 1:nponds
    for k = 1:nponds
        R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
    end
end
for j = i:nponds
    for k = 1:nponds
        if Ad_C(j)~=0
            R_D(j,k) = R_D(j,k)*Ad_C(j);
        end
    end
end
```

```
        R_Dtrans = R_D';
        R_C1trans = R_C1';
        R_C2trans = R_C2';

        % Now we create our immigration and emigration vectors.  We do it
        % by multiplying each row vector of our completed total movement
        % rate matries R_{D,C1,C2}
for h = 1:nponds
            immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
            immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
            immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
        end

        immVectorJuvD = sum(immMatrixJuvD,2);
        immVectorJuvC1 = sum(immMatrixJuvC1,2);
        immVectorJuvC2 = sum(immMatrixJuvC2,2);
        emmVectorJuvD = R_Dtrans*Juv_D;
        emmVectorJuvC1 = R_C1trans*Juv_C1;
        emmVectorJuvC2 = R_C2trans*Juv_C2;

    % The following three paragraphs are used mearly to put all the above
    % immigration/emigration information in vectors of the appropriate size
    % to be subtracted from and added to the population vector.

        x_JuvD_dummy=[0 0 1];
        imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
        emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

        x_JuvC1_dummy=[0 0 0 1 0];
        imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
        emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

        x_JuvC2_dummy=[0 0 0 0 1];
        imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
        emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
```

```
    % the immigration/emigration rules outlined above.
        draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
        catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')+
(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
        neg_d=find(draytonii<0); draytonii(neg_d)=0;
        neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;

% Now we want to save this run's data.
        fid = fopen(['Juv_DAt' num2str(w) 'EffortPondDraining.dat' ],'a');
%fprintf(fid, '%3.6f ', i);
        fprintf(fid, '%3.6f ', Juv_D);fprintf(fid,'\n');
        fclose(fid);

    end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code averages out the time series' created for various levels of
% eradication of bullfrog tadpoles via dip netting or pond draining of
% each of the six ponds we are studying.  It calls 'data' which was
% produced via the code: 'AvePondDraining.m'.  This code produces a figure
% which shows the average CRLF juvenile population size over various levels
% of eradication (0-100% over 5% increments).

% Here we are calling our data:
JuvD0 = load(['Juv_DAt0EffortPondDraining.dat']);
JuvD0_05 = load(['Juv_DAt0.05EffortPondDraining.dat']);
JuvD0_1 = load(['Juv_DAt0.1EffortPondDraining.dat']);
JuvD0_15 = load(['Juv_DAt0.15EffortPondDraining.dat']);
JuvD0_2 = load(['Juv_DAt0.2EffortPondDraining.dat']);
JuvD0_25 = load(['Juv_DAt0.25EffortPondDraining.dat']);
JuvD0_3 = load(['Juv_DAt0.3EffortPondDraining.dat']);
```

```
JuvD0_35 = load(['Juv_DAt0.35EffortPondDraining.dat']);
JuvD0_4 = load(['Juv_DAt0.4EffortPondDraining.dat']);
JuvD0_45 = load(['Juv_DAt0.45EffortPondDraining.dat']);
JuvD0_5 = load(['Juv_DAt0.5EffortPondDraining.dat']);
JuvD0_55 = load(['Juv_DAt0.55EffortPondDraining.dat']);
JuvD0_6 = load(['Juv_DAt0.6EffortPondDraining.dat']);
JuvD0_65 = load(['Juv_DAt0.65EffortPondDraining.dat']);
JuvD0_7 = load(['Juv_DAt0.7EffortPondDraining.dat']);
JuvD0_75 = load(['Juv_DAt0.75EffortPondDraining.dat']);
JuvD0_8 = load(['Juv_DAt0.8EffortPondDraining.dat']);
JuvD0_85 = load(['Juv_DAt0.85EffortPondDraining.dat']);
JuvD0_9 = load(['Juv_DAt0.9EffortPondDraining.dat']);
JuvD0_95 = load(['Juv_DAt0.95EffortPondDraining.dat']);
JuvD1 = load(['Juv_DAt1EffortPondDraining.dat']);

% In this for-loop, we are arranging the data according to pond.  We only
% collect data from the last 100 years of a 200 year simulation in order to
% exclude any transient behavior that may be present in the beginning of
% the simulations.
for n = 101:200

    PondAD(n-100,1)=JuvD0(n,2);
    PondAD(n-100,2)=JuvD0_05(n,2);
    PondAD(n-100,3)=JuvD0_1(n,2);
    PondAD(n-100,4)=JuvD0_15(n,2);
    PondAD(n-100,5)=JuvD0_2(n,2);
    PondAD(n-100,6)=JuvD0_25(n,2);
    PondAD(n-100,7)=JuvD0_3(n,2);
    PondAD(n-100,8)=JuvD0_35(n,2);
    PondAD(n-100,9)=JuvD0_4(n,2);
    PondAD(n-100,10)=JuvD0_45(n,2);
    PondAD(n-100,11)=JuvD0_5(n,2);
    PondAD(n-100,12)=JuvD0_55(n,2);
    PondAD(n-100,13)=JuvD0_6(n,2);
    PondAD(n-100,14)=JuvD0_65(n,2);
    PondAD(n-100,15)=JuvD0_7(n,2);
    PondAD(n-100,16)=JuvD0_75(n,2);
```

```
PondAD(n-100,17)=JuvD0_8(n,2);
PondAD(n-100,18)=JuvD0_85(n,2);
PondAD(n-100,19)=JuvD0_9(n,2);
PondAD(n-100,20)=JuvD0_95(n,2);
PondAD(n-100,21)=JuvD1(n,2);

PondWD(n-100,1)=JuvD0(n,3);
PondWD(n-100,3)=JuvD0_1(n,3);
PondWD(n-100,4)=JuvD0_15(n,3);
PondWD(n-100,5)=JuvD0_2(n,3);
PondWD(n-100,6)=JuvD0_25(n,3);
PondWD(n-100,7)=JuvD0_3(n,3);
PondWD(n-100,8)=JuvD0_35(n,3);
PondWD(n-100,9)=JuvD0_4(n,3);
PondWD(n-100,10)=JuvD0_45(n,3);
PondWD(n-100,11)=JuvD0_5(n,3);
PondWD(n-100,12)=JuvD0_55(n,3);
PondWD(n-100,13)=JuvD0_6(n,3);
PondWD(n-100,14)=JuvD0_65(n,3);
PondWD(n-100,15)=JuvD0_7(n,3);
PondWD(n-100,16)=JuvD0_75(n,3);
PondWD(n-100,17)=JuvD0_8(n,3);
PondWD(n-100,18)=JuvD0_85(n,3);
PondWD(n-100,19)=JuvD0_9(n,3);
PondWD(n-100,20)=JuvD0_95(n,3);
PondWD(n-100,21)=JuvD1(n,3);

PondCP(n-100,1)=JuvD0(n,4);
PondCP(n-100,2)=JuvD0_05(n,4);
PondCP(n-100,3)=JuvD0_1(n,4);
PondCP(n-100,4)=JuvD0_15(n,4);
PondCP(n-100,5)=JuvD0_2(n,4);
PondCP(n-100,6)=JuvD0_25(n,4);
PondCP(n-100,7)=JuvD0_3(n,4);
PondCP(n-100,8)=JuvD0_35(n,4);
PondCP(n-100,9)=JuvD0_4(n,4);
PondCP(n-100,10)=JuvD0_45(n,4);
```

```
PondCP(n-100,11)=JuvD0_5(n,4);
PondCP(n-100,12)=JuvD0_55(n,4);
PondCP(n-100,13)=JuvD0_6(n,4);
PondCP(n-100,14)=JuvD0_65(n,4);
PondCP(n-100,15)=JuvD0_7(n,4);
PondCP(n-100,16)=JuvD0_75(n,4);
PondCP(n-100,17)=JuvD0_8(n,4);
PondCP(n-100,18)=JuvD0_85(n,4);
PondCP(n-100,19)=JuvD0_9(n,4);
PondCP(n-100,20)=JuvD0_95(n,4);
PondCP(n-100,21)=JuvD1(n,4);

PondOT(n-100,1)=JuvD0(n,5);
PondOT(n-100,2)=JuvD0_05(n,5);
PondOT(n-100,3)=JuvD0_1(n,5);
PondOT(n-100,4)=JuvD0_15(n,5);
PondOT(n-100,5)=JuvD0_2(n,5);
PondOT(n-100,6)=JuvD0_25(n,5);
PondOT(n-100,7)=JuvD0_3(n,5);
PondOT(n-100,8)=JuvD0_35(n,5);
PondOT(n-100,9)=JuvD0_4(n,5);
PondOT(n-100,10)=JuvD0_45(n,5);
PondOT(n-100,11)=JuvD0_5(n,5);
PondOT(n-100,12)=JuvD0_55(n,5);
PondOT(n-100,13)=JuvD0_6(n,5);
PondOT(n-100,14)=JuvD0_65(n,5);
PondOT(n-100,15)=JuvD0_7(n,5);
PondOT(n-100,16)=JuvD0_75(n,5);
PondOT(n-100,17)=JuvD0_8(n,5);
PondOT(n-100,18)=JuvD0_85(n,5);
PondOT(n-100,19)=JuvD0_9(n,5);
PondOT(n-100,20)=JuvD0_95(n,5);
PondOT(n-100,21)=JuvD1(n,5);

PondMP(n-100,1)=JuvD0(n,6);
PondMP(n-100,2)=JuvD0_05(n,6);
PondMP(n-100,3)=JuvD0_1(n,6);
```

```
PondMP(n-100,4)=JuvD0_15(n,6);
PondMP(n-100,5)=JuvD0_2(n,6);
PondMP(n-100,6)=JuvD0_25(n,6);
PondMP(n-100,7)=JuvD0_3(n,6);
PondMP(n-100,8)=JuvD0_35(n,6);
PondMP(n-100,9)=JuvD0_4(n,6);
PondMP(n-100,10)=JuvD0_45(n,6);
PondMP(n-100,11)=JuvD0_5(n,6);
PondMP(n-100,12)=JuvD0_55(n,6);
PondMP(n-100,13)=JuvD0_6(n,6);
PondMP(n-100,14)=JuvD0_65(n,6);
PondMP(n-100,15)=JuvD0_7(n,6);
PondMP(n-100,16)=JuvD0_75(n,6);
PondMP(n-100,17)=JuvD0_8(n,6);
PondMP(n-100,18)=JuvD0_85(n,6);
PondMP(n-100,19)=JuvD0_9(n,6);
PondMP(n-100,20)=JuvD0_95(n,6);
PondMP(n-100,21)=JuvD1(n,6);

PondBL(n-100,1)=JuvD0(n,8);
PondBL(n-100,2)=JuvD0_05(n,8);
PondBL(n-100,3)=JuvD0_1(n,8);
PondBL(n-100,4)=JuvD0_15(n,8);
PondBL(n-100,5)=JuvD0_2(n,8);
PondBL(n-100,6)=JuvD0_25(n,8);
PondBL(n-100,7)=JuvD0_3(n,8);
PondBL(n-100,8)=JuvD0_35(n,8);
PondBL(n-100,9)=JuvD0_4(n,8);
PondBL(n-100,10)=JuvD0_45(n,8);
PondBL(n-100,11)=JuvD0_5(n,8);
PondBL(n-100,12)=JuvD0_55(n,8);
PondBL(n-100,13)=JuvD0_6(n,8);
PondBL(n-100,14)=JuvD0_65(n,8);
PondBL(n-100,15)=JuvD0_7(n,8);
PondBL(n-100,16)=JuvD0_75(n,8);
PondBL(n-100,17)=JuvD0_8(n,8);
PondBL(n-100,18)=JuvD0_85(n,8);
```

```
    PondBL(n-100,19)=JuvD0_9(n,8);
    PondBL(n-100,20)=JuvD0_95(n,8);
    PondBL(n-100,21)=JuvD1(n,8);
end

% Calcultating averages and standard deviations:
t=(0:0.05:1);% Percent eradication incemented.
yAD = mean(PondAD,1);
eAD = std(PondAD,1,1);
yWD = mean(PondWD,1);
eWD = std(PondWD,1,1);
yCP = mean(PondCP,1);
eCP = std(PondCP,1,1);
yOT = mean(PondOT,1);
eOT = std(PondOT,1,1);
yMP = mean(PondMP,1);
eMP = std(PondMP,1,1);
yBL = mean(PondBL,1);
eBL = std(PondBL,1,1);

% Producing the figure:
figure
hold on
errorbar(t,yAD,eAD,'og');
errorbar(t,yWD,eWD,'xc');
errorbar(t,yCP,eCP,'xk');
errorbar(t,yOT,eOT,'xm');
errorbar(t,yMP,eMP,'or');
errorbar(t,yBL,eBL,'xb');
title('Management: Dip Netting');
xlabel('Proportion of Bullfrog Tadpoles Erraticated');
ylabel('Average CRLF Juvenile Population');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code simulates the erection and maintanence of drift fences every
```

```
% year for 200 years over various levels of eradication of the bullfrog
% metamorphs (0-100%, at 5% increments). Running the code
% DriftFenceFigure.m will produce the figure.

close all;
clear;

% Number of ponds.

nponds = 8;

% Pond distance matrix initialization.  Called M in the thesis.

Dnbyn = zeros(nponds,nponds);

% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
```

```
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);

Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
```

```
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;

%  Bullfrog max distance.

for i = 1:nponds
    for j = 1:nponds
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
    end
end

% CRLF max distance.

for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
end

% Initial conditions.
u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.

% IC for initializing a single bullfrog.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u';
catesbeiana(:,1) = v';

% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;
just_2nd_year_juv = zeros(nponds,nsteps+1);

% Parameters.
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

% w is level of eradication.
for w = 0:0.05:1
    for i = 1:nsteps
        for j = 1:nponds

% Assigning the constant entries.
            D(j*3-2,3*j) = r*p4;
```

```
            D(3*j,3*j) = p4;
            C(j*5-1,j*5-2) = s3;
            C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.
            if j == 1
                C(j*5-3,j*5-4) = 0;
            elseif j == 2
                C(j*5-3,j*5-4) = 0;
            elseif j == 6
                C(j*5-3,j*5-4) = 0;
            elseif j == 7
                C(j*5-3,j*5-4) = 0;
            else
                C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
            end
            C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
        end

    % Assigning the function entries.
        for j = 1:nponds
            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5,j*5-1) = s4;

        % Applying eradication effort.
            C(j*5-2,j*5-3) = (1-w)*s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = (1-w)*sFT*exp(-mu*catesbeiana(5*j,i));

        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.
            Juv_D(j,1)=draytonii(j*3-1,i);
            Juv_C1(j,1)=catesbeiana(j*5-2,i);
```

```
        Juv_C2(j,1)=catesbeiana(j*5-1,i);

    % CRLF habitat quality indicator calculation.
        Ad_C(j,1) = catesbeiana(j*5,i);
        if Ad_C(j)==0
            Ad_C(j)=1;
        end
        if Ad_C(j)<1
            Ad_C(j)=1;
        else
            Ad_C(j)=1-(1/(Ad_C(j)));
        end
        if Ad_C(j)==0
            Ad_C(j)=1;
        end
    end

% Here we create our gamma distributions which tells us what proportion
% of the juvenile population will move to which pond.  Note that since
% this is the 'stochastic' version of the simulation, we update the
% gamma distribution every year (time step) for each pond.
% Furthermore, each updated gamma distribution is dependent upon the
% source pond's juvenile population.  Distance values are drawn and
% used in their corresponding gamma distribution to obtain rates of
% dispersion (immigration out of a pond).
    [R_D]=calculate_rij_stoch_Dnew(DnbynD,Juv_D);
    [R_C1]=calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
    [R_C2]=calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

% Stage specific choice probability calculation.
    newDnbynC2 = zeros(nponds,nponds);
    RecipnewDnbynC2 = zeros(nponds,nponds);
    [rowC2,colC2] = find(R_C2);
    DispDepInflC2 = zeros(nponds,nponds);
    for g = 1:length(rowC2)
            newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
    end
```

```
for h = 1:nponds

    if nnz(R_D(h,:)) > 1
        R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
    end
    if nnz(R_C1(h,:)) > 1
        R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
    end
    for k = 1:nponds
        if newDnbynC2(h,k)==0
            DispDepInflC2(h,k) = 0;
        else
            RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);
        end
    end
end
for j = 1:length(rowC2)
        if colC2(j)==1
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==2
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==6
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==7
            R_C2(rowC2(j),colC2(j))=0;
        end
end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
end
for j = 1:nponds
    for k = 1:nponds
```

```
            R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
        end
    end
    for j = i:nponds
        for k = 1:nponds
            if Ad_C(j)~=0
                R_D(j,k) = R_D(j,k)*Ad_C(j);
            end
        end
    end
    R_Dtrans = R_D';
    R_C1trans = R_C1';
    R_C2trans = R_C2';

    % Now that we have our total movement probability matrices
    % R_{D,C1,C2}, we can now create our immigration vectors by
    % multipying each row of the R's by each pond's distinct juvenile
    % population size.

    for h = 1:nponds
        immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
        immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
        immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
    end

    immVectorJuvD = sum(immMatrixJuvD,2);
    immVectorJuvC1 = sum(immMatrixJuvC1,2);
    immVectorJuvC2 = sum(immMatrixJuvC2,2);

    % Emigration vectors are calcultated by multipying the traspose of
    % the R's by their corresponding juvenile population vector.
    emmVectorJuvD = R_Dtrans*Juv_D;
    emmVectorJuvC1 = R_C1trans*Juv_C1;
    emmVectorJuvC2 = R_C2trans*Juv_C2;

% The following three paragraphs are used mearly to put all the above
% immigration/emigration information in vectors of the appropriate size
```

```
    % to be subtracted from and added to the population vector.
        x_JuvD_dummy=[0 0 1];
        imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
        emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

        x_JuvC1_dummy=[0 0 0 1 0];
        imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
        emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

        x_JuvC2_dummy=[0 0 0 0 1];
        imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
        emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
    % the immigration/emigration rules outlined above.
draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
        catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')
+(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
        neg_d=find(draytonii<0); draytonii(neg_d)=0;
        neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;

 % Here we are saving the data from this simulation run.
        fid = fopen(['Juv_DAt' num2str(w) 'EffortDriftFence.dat' ],'a');
%fprintf(fid, '%3.6f ', i);
        fprintf(fid, '%3.6f ', Juv_D);fprintf(fid,'\n');
        fclose(fid);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code averages out the time series' created for various levels of
% eradication of metamorphic bullfrogs via the erection of a drift fence
% around the shorline of each of the six ponds we are studying.  It calls
```

```
% ’data’ which was produced via the code: ’AveDriftFence.m’.  This code
% produces a figure which shows the average CRLF juvenile population size
% over various levels of eradication (0-100% over 5% increments).

% Here we are calling our data:
JuvD0 = load([’Juv_DAt0EffortDriftFence.dat’]);
JuvD0_05 = load([’Juv_DAt0.05EffortDriftFence.dat’]);
JuvD0_1 = load([’Juv_DAt0.1EffortDriftFence.dat’]);
JuvD0_15 = load([’Juv_DAt0.15EffortDriftFence.dat’]);
JuvD0_2 = load([’Juv_DAt0.2EffortDriftFence.dat’]);
JuvD0_25 = load([’Juv_DAt0.25EffortDriftFence.dat’]);
JuvD0_3 = load([’Juv_DAt0.3EffortDriftFence.dat’]);
JuvD0_35 = load([’Juv_DAt0.35EffortDriftFence.dat’]);
JuvD0_4 = load([’Juv_DAt0.4EffortDriftFence.dat’]);
JuvD0_45 = load([’Juv_DAt0.45EffortDriftFence.dat’]);
JuvD0_5 = load([’Juv_DAt0.5EffortDriftFence.dat’]);
JuvD0_55 = load([’Juv_DAt0.55EffortDriftFence.dat’]);
JuvD0_6 = load([’Juv_DAt0.6EffortDriftFence.dat’]);
JuvD0_65 = load([’Juv_DAt0.65EffortDriftFence.dat’]);
JuvD0_7 = load([’Juv_DAt0.7EffortDriftFence.dat’]);
JuvD0_75 = load([’Juv_DAt0.75EffortDriftFence.dat’]);
JuvD0_8 = load([’Juv_DAt0.8EffortDriftFence.dat’]);
JuvD0_85 = load([’Juv_DAt0.85EffortDriftFence.dat’]);
JuvD0_9 = load([’Juv_DAt0.9EffortDriftFence.dat’]);
JuvD0_95 = load([’Juv_DAt0.95EffortDriftFence.dat’]);
JuvD1 = load([’Juv_DAt1EffortDriftFence.dat’]);

% In this for-loop, we are arranging the data according to pond.  We only
% collect data from the last 100 years of a 200 year simulation in order to
% exclude any transient behavior that may be present in the beginning of
% the simulations.
for n = 101:200

    PondAD(n-100,1)=JuvD0(n,2);
    PondAD(n-100,2)=JuvD0_05(n,2);
    PondAD(n-100,3)=JuvD0_1(n,2);
    PondAD(n-100,4)=JuvD0_15(n,2);
```

```
PondAD(n-100,5)=JuvD0_2(n,2);
PondAD(n-100,6)=JuvD0_25(n,2);
PondAD(n-100,7)=JuvD0_3(n,2);
PondAD(n-100,8)=JuvD0_35(n,2);
PondAD(n-100,9)=JuvD0_4(n,2);
PondAD(n-100,10)=JuvD0_45(n,2);
PondAD(n-100,11)=JuvD0_5(n,2);
PondAD(n-100,12)=JuvD0_55(n,2);
PondAD(n-100,13)=JuvD0_6(n,2);
PondAD(n-100,14)=JuvD0_65(n,2);
PondAD(n-100,15)=JuvD0_7(n,2);
PondAD(n-100,16)=JuvD0_75(n,2);
PondAD(n-100,17)=JuvD0_8(n,2);
PondAD(n-100,18)=JuvD0_85(n,2);
PondAD(n-100,19)=JuvD0_9(n,2);
PondAD(n-100,20)=JuvD0_95(n,2);
PondAD(n-100,21)=JuvD1(n,2);

PondWD(n-100,1)=JuvD0(n,3);
PondWD(n-100,3)=JuvD0_1(n,3);
PondWD(n-100,4)=JuvD0_15(n,3);
PondWD(n-100,5)=JuvD0_2(n,3);
PondWD(n-100,6)=JuvD0_25(n,3);
PondWD(n-100,7)=JuvD0_3(n,3);
PondWD(n-100,8)=JuvD0_35(n,3);
PondWD(n-100,9)=JuvD0_4(n,3);
PondWD(n-100,10)=JuvD0_45(n,3);
PondWD(n-100,11)=JuvD0_5(n,3);
PondWD(n-100,12)=JuvD0_55(n,3);
PondWD(n-100,13)=JuvD0_6(n,3);
PondWD(n-100,14)=JuvD0_65(n,3);
PondWD(n-100,15)=JuvD0_7(n,3);
PondWD(n-100,16)=JuvD0_75(n,3);
PondWD(n-100,17)=JuvD0_8(n,3);
PondWD(n-100,18)=JuvD0_85(n,3);
PondWD(n-100,19)=JuvD0_9(n,3);
PondWD(n-100,20)=JuvD0_95(n,3);
```

```
PondWD(n-100,21)=JuvD1(n,3);

PondCP(n-100,1)=JuvD0(n,4);
PondCP(n-100,2)=JuvD0_05(n,4);
PondCP(n-100,3)=JuvD0_1(n,4);
PondCP(n-100,4)=JuvD0_15(n,4);
PondCP(n-100,5)=JuvD0_2(n,4);
PondCP(n-100,6)=JuvD0_25(n,4);
PondCP(n-100,7)=JuvD0_3(n,4);
PondCP(n-100,8)=JuvD0_35(n,4);
PondCP(n-100,9)=JuvD0_4(n,4);
PondCP(n-100,10)=JuvD0_45(n,4);
PondCP(n-100,11)=JuvD0_5(n,4);
PondCP(n-100,12)=JuvD0_55(n,4);
PondCP(n-100,13)=JuvD0_6(n,4);
PondCP(n-100,14)=JuvD0_65(n,4);
PondCP(n-100,15)=JuvD0_7(n,4);
PondCP(n-100,16)=JuvD0_75(n,4);
PondCP(n-100,17)=JuvD0_8(n,4);
PondCP(n-100,18)=JuvD0_85(n,4);
PondCP(n-100,19)=JuvD0_9(n,4);
PondCP(n-100,20)=JuvD0_95(n,4);
PondCP(n-100,21)=JuvD1(n,4);

PondOT(n-100,1)=JuvD0(n,5);
PondOT(n-100,2)=JuvD0_05(n,5);
PondOT(n-100,3)=JuvD0_1(n,5);
PondOT(n-100,4)=JuvD0_15(n,5);
PondOT(n-100,5)=JuvD0_2(n,5);
PondOT(n-100,6)=JuvD0_25(n,5);
PondOT(n-100,7)=JuvD0_3(n,5);
PondOT(n-100,8)=JuvD0_35(n,5);
PondOT(n-100,9)=JuvD0_4(n,5);
PondOT(n-100,10)=JuvD0_45(n,5);
PondOT(n-100,11)=JuvD0_5(n,5);
PondOT(n-100,12)=JuvD0_55(n,5);
PondOT(n-100,13)=JuvD0_6(n,5);
```

```
PondOT(n-100,14)=JuvD0_65(n,5);
PondOT(n-100,15)=JuvD0_7(n,5);
PondOT(n-100,16)=JuvD0_75(n,5);
PondOT(n-100,17)=JuvD0_8(n,5);
PondOT(n-100,18)=JuvD0_85(n,5);
PondOT(n-100,19)=JuvD0_9(n,5);
PondOT(n-100,20)=JuvD0_95(n,5);
PondOT(n-100,21)=JuvD1(n,5);

PondMP(n-100,1)=JuvD0(n,6);
PondMP(n-100,2)=JuvD0_05(n,6);
PondMP(n-100,3)=JuvD0_1(n,6);
PondMP(n-100,4)=JuvD0_15(n,6);
PondMP(n-100,5)=JuvD0_2(n,6);
PondMP(n-100,6)=JuvD0_25(n,6);
PondMP(n-100,7)=JuvD0_3(n,6);
PondMP(n-100,8)=JuvD0_35(n,6);
PondMP(n-100,9)=JuvD0_4(n,6);
PondMP(n-100,10)=JuvD0_45(n,6);
PondMP(n-100,11)=JuvD0_5(n,6);
PondMP(n-100,12)=JuvD0_55(n,6);
PondMP(n-100,13)=JuvD0_6(n,6);
PondMP(n-100,14)=JuvD0_65(n,6);
PondMP(n-100,15)=JuvD0_7(n,6);
PondMP(n-100,16)=JuvD0_75(n,6);
PondMP(n-100,17)=JuvD0_8(n,6);
PondMP(n-100,18)=JuvD0_85(n,6);
PondMP(n-100,19)=JuvD0_9(n,6);
PondMP(n-100,20)=JuvD0_95(n,6);
PondMP(n-100,21)=JuvD1(n,6);

PondBL(n-100,1)=JuvD0(n,8);
PondBL(n-100,2)=JuvD0_05(n,8);
PondBL(n-100,3)=JuvD0_1(n,8);
PondBL(n-100,4)=JuvD0_15(n,8);
PondBL(n-100,5)=JuvD0_2(n,8);
PondBL(n-100,6)=JuvD0_25(n,8);
```

```
    PondBL(n-100,7)=JuvD0_3(n,8);
    PondBL(n-100,8)=JuvD0_35(n,8);
    PondBL(n-100,9)=JuvD0_4(n,8);
    PondBL(n-100,10)=JuvD0_45(n,8);
    PondBL(n-100,11)=JuvD0_5(n,8);
    PondBL(n-100,12)=JuvD0_55(n,8);
    PondBL(n-100,13)=JuvD0_6(n,8);
    PondBL(n-100,14)=JuvD0_65(n,8);
    PondBL(n-100,15)=JuvD0_7(n,8);
    PondBL(n-100,16)=JuvD0_75(n,8);
    PondBL(n-100,17)=JuvD0_8(n,8);
    PondBL(n-100,18)=JuvD0_85(n,8);
    PondBL(n-100,19)=JuvD0_9(n,8);
    PondBL(n-100,20)=JuvD0_95(n,8);
    PondBL(n-100,21)=JuvD1(n,8);
end

% Calcultating averages and standard deviations:
t=(0:0.05:1);% Percent eradication incemented.
yAD = mean(PondAD,1);
eAD = std(PondAD,1,1);
yWD = mean(PondWD,1);
eWD = std(PondWD,1,1);
yCP = mean(PondCP,1);
eCP = std(PondCP,1,1);
yOT = mean(PondOT,1);
eOT = std(PondOT,1,1);
yMP = mean(PondMP,1);
eMP = std(PondMP,1,1);
yBL = mean(PondBL,1);
eBL = std(PondBL,1,1);

% Producing the figure:
figure
hold on
errorbar(t,yAD,eAD,'og');
errorbar(t,yWD,eWD,'xc');
```

```
errorbar(t,yCP,eCP,'xk');
errorbar(t,yOT,eOT,'xm');
errorbar(t,yMP,eMP,'or');
errorbar(t,yBL,eBL,'xb');
title('Management: Drift Fence');
xlabel('Proportion of Bullfrog Metamorphs Erraticated');
ylabel('Average CRLF Juvenile Population');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code runs a simulation in which all terrestrial bullfrogs are shoot
% at various levels of eradication (0-100% at 5% increments) every year for
% 200 years.  The data will be saved for the code ShootingFigure.m to
% produce a figure.

close all;
clear;

% Number of ponds.

nponds = 8;

% Pond distance matrix initialization.  In thesis it's called M.

Dnbyn = zeros(nponds,nponds);

% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
```

```
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);

Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
```

```
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;

% Bullfrog max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
    end
end

% CRLF max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
```

```
end


% Initial conditions.
u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.

% IC for initializing a single bullfrog.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u';
catesbeiana(:,1) = v';

% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;

% Parameters.
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
```

```
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence


% w is the level of eradication.
for w = 0:0.05:1
    for i = 1:nsteps
        for j = 1:nponds
        % Assigning the constant entries.
            D(j*3-2,3*j) = r*p4;
            D(3*j,3*j) = p4;
            C(j*5-1,j*5-2) = s3;
            C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.
            if j == 1
                C(j*5-3,j*5-4) = 0;
            elseif j == 2
                C(j*5-3,j*5-4) = 0;
            elseif j == 6
                C(j*5-3,j*5-4) = 0;
            elseif j == 7
                C(j*5-3,j*5-4) = 0;
            else
                C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
            end
            C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
        end

    % Assigning the function entries.
        for j = 1:nponds
            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
```

```
            C(j*5,j*5-1) = s4;

% Applying eradication effort.
            C(j*5-1,j*5-2) = (1-w)*s3;
            C(j*5,j*5-1) = (1-w)*s4;
            C(j*5,j*5) = (1-w)*s5;

        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.
            Juv_D(j,1)=draytonii(j*3-1,i);
            Juv_C1(j,1)=catesbeiana(j*5-2,i);
            Juv_C2(j,1)=catesbeiana(j*5-1,i);

        % Implementing CRLF Habitat quaility indicator.
            Ad_C(j,1) = catesbeiana(j*5,i);
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
            if Ad_C(j)<1
                Ad_C(j)=1;
            else
                Ad_C(j)=1-(1/(Ad_C(j)));
            end
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
        end

    % Here we create our gamma distributions which tells us what proportion
    % of the juvenile population will move to which pond.  Note that since
    % this is the 'stochastic' version of the simulation, we update the
    % gamma distribution every year (time step) for each pond.
    % Furthermore, each updated gamma distribution is dependent upon the
    % source pond's juvenile population.  Distance values are drawn and
    % used in their corresponding gamma distribution to obtain rates of
    % dispersion (immigration out of a pond).
        [R_D]=calculate_rij_stoch_Dnew(DnbynD,Juv_D);
```

```
    [R_C1]=calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
    [R_C2]=calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

% Here we have the stage specific movment rules running.
    newDnbynC2 = zeros(nponds,nponds);
    RecipnewDnbynC2 = zeros(nponds,nponds);
    [rowC2,colC2] = find(R_C2);
    DispDepInflC2 = zeros(nponds,nponds);
    for g = 1:length(rowC2)
            newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
    end

    for h = 1:nponds

        if nnz(R_D(h,:)) > 1
            R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
        end
        if nnz(R_C1(h,:)) > 1
            R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
        end
               for k = 1:nponds
            if newDnbynC2(h,k)==0
                DispDepInflC2(h,k) = 0;
            else
                RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);
            end
        end
    end
    for j = 1:length(rowC2)
            if colC2(j)==1
                R_C2(rowC2(j),colC2(j))=0;
            elseif colC2(j)==2
                R_C2(rowC2(j),colC2(j))=0;
            elseif colC2(j)==6
                R_C2(rowC2(j),colC2(j))=0;
            elseif colC2(j)==7
                R_C2(rowC2(j),colC2(j))=0;
```

```
        end
end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
end
for j = 1:nponds
    for k = 1:nponds
        R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
    end
end
for j = i:nponds
    for k = 1:nponds
        if Ad_C(j)~=0
            R_D(j,k) = R_D(j,k)*Ad_C(j);
        end
    end
end
R_Dtrans = R_D';
R_C1trans = R_C1';
R_C2trans = R_C2';

% Here we create the immigration vectors by multiplying each row of
% the R matrices by it's corresponding juvenile population vector
% entry.
for h = 1:nponds
    immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
    immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
    immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
end

immVectorJuvD = sum(immMatrixJuvD,2);
immVectorJuvC1 = sum(immMatrixJuvC1,2);
immVectorJuvC2 = sum(immMatrixJuvC2,2);
```

```
        % Emigration vector calculation.
        emmVectorJuvD = R_Dtrans*Juv_D;
        emmVectorJuvC1 = R_C1trans*Juv_C1;
        emmVectorJuvC2 = R_C2trans*Juv_C2;

    % The following three paragraphs are used mearly to put all the above
    % immigration/emigration information in vectors of the appropriate size
    % to be subtracted from and added to the population vector.
        x_JuvD_dummy=[0 0 1];
        imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
        emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

        x_JuvC1_dummy=[0 0 0 1 0];
        imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
        emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

        x_JuvC2_dummy=[0 0 0 0 1];
        imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
        emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
    % the immigration/emigration rules outlined above.
        draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
        catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')
+(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
        neg_d=find(draytonii<0); draytonii(neg_d)=0;
        neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;


        % This bit saves the data we need after each simulation run.
        fid = fopen(['Juv_DAt' num2str(w) 'EffortShooting.dat' ],'a');
%fprintf(fid, '%3.6f ', i);
        fprintf(fid, '%3.6f ', Juv_D);fprintf(fid,'\n');
```

```
        fclose(fid);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code averages out the time series' created for various levels of
% eradication of terrestrial bullfrogs via shooting at each of the six
% ponds we are studying.  It calls 'data' which was produced via the code:
% 'AveShooting.m'.  This code produces a figure which shows the average
% CRLF juvenile population size over various levels of eradication (0-100%
% over 5% increments).

% Here we are calling our data:
JuvD0 = load(['Juv_DAt0EffortShooting.dat']);
JuvD0_05 = load(['Juv_DAt0.05EffortShooting.dat']);
JuvD0_1 = load(['Juv_DAt0.1EffortShooting.dat']);
JuvD0_15 = load(['Juv_DAt0.15EffortShooting.dat']);
JuvD0_2 = load(['Juv_DAt0.2EffortShooting.dat']);
JuvD0_25 = load(['Juv_DAt0.25EffortShooting.dat']);
JuvD0_3 = load(['Juv_DAt0.3EffortShooting.dat']);
JuvD0_35 = load(['Juv_DAt0.35EffortShooting.dat']);
JuvD0_4 = load(['Juv_DAt0.4EffortShooting.dat']);
JuvD0_45 = load(['Juv_DAt0.45EffortShooting.dat']);
JuvD0_5 = load(['Juv_DAt0.5EffortShooting.dat']);
JuvD0_55 = load(['Juv_DAt0.55EffortShooting.dat']);
JuvD0_6 = load(['Juv_DAt0.6EffortShooting.dat']);
JuvD0_65 = load(['Juv_DAt0.65EffortShooting.dat']);
JuvD0_7 = load(['Juv_DAt0.7EffortShooting.dat']);
JuvD0_75 = load(['Juv_DAt0.75EffortShooting.dat']);
JuvD0_8 = load(['Juv_DAt0.8EffortShooting.dat']);
JuvD0_85 = load(['Juv_DAt0.85EffortShooting.dat']);
JuvD0_9 = load(['Juv_DAt0.9EffortShooting.dat']);
JuvD0_95 = load(['Juv_DAt0.95EffortShooting.dat']);
JuvD1 = load(['Juv_DAt1EffortShooting.dat']);

% In this for-loop, we are arranging the data according to pond.  We only
```

```
% collect data from the last 100 years of a 200 year simulation in order to
% exclude any transient behavior that may be present in the beginning of
% the simulations.
for n = 101:200

    PondAD(n-100,1)=JuvD0(n,2);
    PondAD(n-100,2)=JuvD0_05(n,2);
    PondAD(n-100,3)=JuvD0_1(n,2);
    PondAD(n-100,4)=JuvD0_15(n,2);
    PondAD(n-100,5)=JuvD0_2(n,2);
    PondAD(n-100,6)=JuvD0_25(n,2);
    PondAD(n-100,7)=JuvD0_3(n,2);
    PondAD(n-100,8)=JuvD0_35(n,2);
    PondAD(n-100,9)=JuvD0_4(n,2);
    PondAD(n-100,10)=JuvD0_45(n,2);
    PondAD(n-100,11)=JuvD0_5(n,2);
    PondAD(n-100,12)=JuvD0_55(n,2);
    PondAD(n-100,13)=JuvD0_6(n,2);
    PondAD(n-100,14)=JuvD0_65(n,2);
    PondAD(n-100,15)=JuvD0_7(n,2);
    PondAD(n-100,16)=JuvD0_75(n,2);
    PondAD(n-100,17)=JuvD0_8(n,2);
    PondAD(n-100,18)=JuvD0_85(n,2);
    PondAD(n-100,19)=JuvD0_9(n,2);
    PondAD(n-100,20)=JuvD0_95(n,2);
    PondAD(n-100,21)=JuvD1(n,2);

    PondWD(n-100,1)=JuvD0(n,3);
    PondWD(n-100,3)=JuvD0_1(n,3);
    PondWD(n-100,4)=JuvD0_15(n,3);
    PondWD(n-100,5)=JuvD0_2(n,3);
    PondWD(n-100,6)=JuvD0_25(n,3);
    PondWD(n-100,7)=JuvD0_3(n,3);
    PondWD(n-100,8)=JuvD0_35(n,3);
    PondWD(n-100,9)=JuvD0_4(n,3);
    PondWD(n-100,10)=JuvD0_45(n,3);
    PondWD(n-100,11)=JuvD0_5(n,3);
```

```
PondWD(n-100,12)=JuvD0_55(n,3);
PondWD(n-100,13)=JuvD0_6(n,3);
PondWD(n-100,14)=JuvD0_65(n,3);
PondWD(n-100,15)=JuvD0_7(n,3);
PondWD(n-100,16)=JuvD0_75(n,3);
PondWD(n-100,17)=JuvD0_8(n,3);
PondWD(n-100,18)=JuvD0_85(n,3);
PondWD(n-100,19)=JuvD0_9(n,3);
PondWD(n-100,20)=JuvD0_95(n,3);
PondWD(n-100,21)=JuvD1(n,3);

PondCP(n-100,1)=JuvD0(n,4);
PondCP(n-100,2)=JuvD0_05(n,4);
PondCP(n-100,3)=JuvD0_1(n,4);
PondCP(n-100,4)=JuvD0_15(n,4);
PondCP(n-100,5)=JuvD0_2(n,4);
PondCP(n-100,6)=JuvD0_25(n,4);
PondCP(n-100,7)=JuvD0_3(n,4);
PondCP(n-100,8)=JuvD0_35(n,4);
PondCP(n-100,9)=JuvD0_4(n,4);
PondCP(n-100,10)=JuvD0_45(n,4);
PondCP(n-100,11)=JuvD0_5(n,4);
PondCP(n-100,12)=JuvD0_55(n,4);
PondCP(n-100,13)=JuvD0_6(n,4);
PondCP(n-100,14)=JuvD0_65(n,4);
PondCP(n-100,15)=JuvD0_7(n,4);
PondCP(n-100,16)=JuvD0_75(n,4);
PondCP(n-100,17)=JuvD0_8(n,4);
PondCP(n-100,18)=JuvD0_85(n,4);
PondCP(n-100,19)=JuvD0_9(n,4);
PondCP(n-100,20)=JuvD0_95(n,4);
PondCP(n-100,21)=JuvD1(n,4);

PondOT(n-100,1)=JuvD0(n,5);
PondOT(n-100,2)=JuvD0_05(n,5);
PondOT(n-100,3)=JuvD0_1(n,5);
PondOT(n-100,4)=JuvD0_15(n,5);
```

```
PondOT(n-100,5)=JuvD0_2(n,5);
PondOT(n-100,6)=JuvD0_25(n,5);
PondOT(n-100,7)=JuvD0_3(n,5);
PondOT(n-100,8)=JuvD0_35(n,5);
PondOT(n-100,9)=JuvD0_4(n,5);
PondOT(n-100,10)=JuvD0_45(n,5);
PondOT(n-100,11)=JuvD0_5(n,5);
PondOT(n-100,12)=JuvD0_55(n,5);
PondOT(n-100,13)=JuvD0_6(n,5);
PondOT(n-100,14)=JuvD0_65(n,5);
PondOT(n-100,15)=JuvD0_7(n,5);
PondOT(n-100,16)=JuvD0_75(n,5);
PondOT(n-100,17)=JuvD0_8(n,5);
PondOT(n-100,18)=JuvD0_85(n,5);
PondOT(n-100,19)=JuvD0_9(n,5);
PondOT(n-100,20)=JuvD0_95(n,5);
PondOT(n-100,21)=JuvD1(n,5);

PondMP(n-100,1)=JuvD0(n,6);
PondMP(n-100,2)=JuvD0_05(n,6);
PondMP(n-100,3)=JuvD0_1(n,6);
PondMP(n-100,4)=JuvD0_15(n,6);
PondMP(n-100,5)=JuvD0_2(n,6);
PondMP(n-100,6)=JuvD0_25(n,6);
PondMP(n-100,7)=JuvD0_3(n,6);
PondMP(n-100,8)=JuvD0_35(n,6);
PondMP(n-100,9)=JuvD0_4(n,6);
PondMP(n-100,10)=JuvD0_45(n,6);
PondMP(n-100,11)=JuvD0_5(n,6);
PondMP(n-100,12)=JuvD0_55(n,6);
PondMP(n-100,13)=JuvD0_6(n,6);
PondMP(n-100,14)=JuvD0_65(n,6);
PondMP(n-100,15)=JuvD0_7(n,6);
PondMP(n-100,16)=JuvD0_75(n,6);
PondMP(n-100,17)=JuvD0_8(n,6);
PondMP(n-100,18)=JuvD0_85(n,6);
PondMP(n-100,19)=JuvD0_9(n,6);
```

```
    PondMP(n-100,20)=JuvD0_95(n,6);
    PondMP(n-100,21)=JuvD1(n,6);

    PondBL(n-100,1)=JuvD0(n,8);
    PondBL(n-100,2)=JuvD0_05(n,8);
    PondBL(n-100,3)=JuvD0_1(n,8);
    PondBL(n-100,4)=JuvD0_15(n,8);
    PondBL(n-100,5)=JuvD0_2(n,8);
    PondBL(n-100,6)=JuvD0_25(n,8);
    PondBL(n-100,7)=JuvD0_3(n,8);
    PondBL(n-100,8)=JuvD0_35(n,8);
    PondBL(n-100,9)=JuvD0_4(n,8);
    PondBL(n-100,10)=JuvD0_45(n,8);
    PondBL(n-100,11)=JuvD0_5(n,8);
    PondBL(n-100,12)=JuvD0_55(n,8);
    PondBL(n-100,13)=JuvD0_6(n,8);
    PondBL(n-100,14)=JuvD0_65(n,8);
    PondBL(n-100,15)=JuvD0_7(n,8);
    PondBL(n-100,16)=JuvD0_75(n,8);
    PondBL(n-100,17)=JuvD0_8(n,8);
    PondBL(n-100,18)=JuvD0_85(n,8);
    PondBL(n-100,19)=JuvD0_9(n,8);
    PondBL(n-100,20)=JuvD0_95(n,8);
    PondBL(n-100,21)=JuvD1(n,8);
end

% Calcultating averages and standard deviations:
t=(0:0.05:1);  % Percent eradication incemented.
yAD = mean(PondAD,1);
eAD = std(PondAD,1,1);
yWD = mean(PondWD,1);
eWD = std(PondWD,1,1);
yCP = mean(PondCP,1);
eCP = std(PondCP,1,1);
yOT = mean(PondOT,1);
eOT = std(PondOT,1,1);
yMP = mean(PondMP,1);
```

```
eMP = std(PondMP,1,1);
yBL = mean(PondBL,1);
eBL = std(PondBL,1,1);

% Producing the figure:
figure
hold on
errorbar(t,yAD,eAD,'og');
errorbar(t,yWD,eWD,'xc');
errorbar(t,yCP,eCP,'xk');
errorbar(t,yOT,eOT,'xm');
errorbar(t,yMP,eMP,'or');
errorbar(t,yBL,eBL,'xb');
title('Management: Shooting');
xlabel('Proportion of Bullfrogs Erraticated');
ylabel('Average CRLF Juvenile Population');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% In this simulation, we assume ponds are drained every year, then every
% other year, then every two years and so on until there are 15 years
% between management applications.  Each simulation runs for 200 years and
% info is stored in order to create a figure using PondDrainingSkipFigure.m


close all;
clear;

% Number of ponds.

nponds = 8;

% Pond distance matrix initialization.  Called M in thesis.

Dnbyn = zeros(nponds,nponds);
```

```
% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);

Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
```

```
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;

% Bullfrog max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
```

```
    end
end

% CRLF max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
end

% Initial conditions.
u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.

% Single bullfrog initialization IC.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u';
catesbeiana(:,1) = v';

% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;
```

```
% Parameters.
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

% f-1 is the number of years skipped.
for f = 1:1:16
    for i = 1:nsteps
        for j = 1:nponds
        % Assigning the constant entries.
            D(j*3-2,3*j) = r*p4;
            D(3*j,3*j) = p4;
            C(j*5-1,j*5-2) = s3;
            C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.
            if j == 1
                C(j*5-3,j*5-4) = 0;
            elseif j == 2
                C(j*5-3,j*5-4) = 0;
            elseif j == 6
                C(j*5-3,j*5-4) = 0;
            elseif j == 7
                C(j*5-3,j*5-4) = 0;
            else
                C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
            end
            C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
        end
```

```
    % Assigning the function entries.
        for j = 1:nponds
            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5,j*5-1) = s4;
        if rem(i,f)==0
            C(j*5-4,j*5) = 0;
            C(j*5-3,j*5-4) = 0;
        end

        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.
            Juv_D(j,1)=draytonii(j*3-1,i);
            Juv_C1(j,1)=catesbeiana(j*5-2,i);
            Juv_C2(j,1)=catesbeiana(j*5-1,i);

            % Here we implement the CRLF habitat quality indicator.
            Ad_C(j,1) = catesbeiana(j*5,i);
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
            if Ad_C(j)<1
                Ad_C(j)=1;
            else
                Ad_C(j)=1-(1/(Ad_C(j)));
            end
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
        end

    % Here we create our gamma distributions which tells us what proportion
    % of the juvenile population will move to which pond.  Note that since
```

```
% this is the 'stochastic' version of the simulation, we update the
% gamma distribution every year (time step) for each pond.
% Furthermore, each updated gamma distribution is dependent upon the
% source pond's juvenile population.  Distance values are drawn and
% used in their corresponding gamma distribution to obtain rates of
% dispersion (immigration out of a pond).
    [R_D]=calculate_rij_stoch_Dnew(DnbynD,Juv_D);
    [R_C1]=calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
    [R_C2]=calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

% Stage specific choice movement probabilites.
    newDnbynC2 = zeros(nponds,nponds);
    RecipnewDnbynC2 = zeros(nponds,nponds);
    [rowC2,colC2] = find(R_C2);
    DispDepInflC2 = zeros(nponds,nponds);
    for g = 1:length(rowC2)
            newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
    end

    for h = 1:nponds

        if nnz(R_D(h,:)) > 1
            R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
        end
        if nnz(R_C1(h,:)) > 1
            R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
        end
        for k = 1:nponds
            if newDnbynC2(h,k)==0
                DispDepInflC2(h,k) = 0;
            else
                RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);
            end
        end
    end
    for j = 1:length(rowC2)
            if colC2(j)==1
```

```
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==2
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==6
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==7
            R_C2(rowC2(j),colC2(j))=0;
        end
end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
end
for j = 1:nponds
    for k = 1:nponds
        R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
    end
end
for j = i:nponds
    for k = 1:nponds
        if Ad_C(j)~=0
            R_D(j,k) = R_D(j,k)*Ad_C(j);
        end
    end
end
R_Dtrans = R_D';
R_C1trans = R_C1';
R_C2trans = R_C2';

% Immigration vector calculation.
for h = 1:nponds
    immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
    immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
    immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
```

```
        end

        immVectorJuvD = sum(immMatrixJuvD,2);
        immVectorJuvC1 = sum(immMatrixJuvC1,2);
        immVectorJuvC2 = sum(immMatrixJuvC2,2);

        % Emigration vector caclulation.
        emmVectorJuvD = R_Dtrans*Juv_D;
        emmVectorJuvC1 = R_C1trans*Juv_C1;
        emmVectorJuvC2 = R_C2trans*Juv_C2;

    % The following three paragraphs are used mearly to put all the above
    % immigration/emigration information in vectors of the appropriate size
    % to be subtracted from and added to the population vector.
        x_JuvD_dummy=[0 0 1];
        imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
        emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

        x_JuvC1_dummy=[0 0 0 1 0];
        imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
        emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

        x_JuvC2_dummy=[0 0 0 0 1];
        imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
        emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
    % the immigration/emigration rules outlined above.
        draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
        catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')
+(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
        neg_d=find(draytonii<0); draytonii(neg_d)=0;
        neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;
```

```
        % Here we save the data every time this simulation runs.
        fid = fopen(['Juv_DSkipping' num2str(f-1) 'YearsPondDraining.dat' ],'a');
 %fprintf(fid, '%3.6f ', i);
        fprintf(fid, '%3.6f ', Juv_D);fprintf(fid,'\n');
        fclose(fid);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code makes a figure using data collected by running
% SkipPondDraining.m

% Calling files.
Skip0Year = load('Juv_DSkipping0YearsPondDraining.dat');
Skip1Year = load('Juv_DSkipping1YearsPondDraining.dat');
Skip2Year = load('Juv_DSkipping2YearsPondDraining.dat');
Skip3Year = load('Juv_DSkipping3YearsPondDraining.dat');
Skip4Year = load('Juv_DSkipping4YearsPondDraining.dat');
Skip5Year = load('Juv_DSkipping5YearsPondDraining.dat');
Skip6Year = load('Juv_DSkipping6YearsPondDraining.dat');
Skip7Year = load('Juv_DSkipping7YearsPondDraining.dat');
Skip8Year = load('Juv_DSkipping8YearsPondDraining.dat');
Skip9Year = load('Juv_DSkipping9YearsPondDraining.dat');
Skip10Year = load('Juv_DSkipping10YearsPondDraining.dat');
Skip11Year = load('Juv_DSkipping11YearsPondDraining.dat');
Skip12Year = load('Juv_DSkipping12YearsPondDraining.dat');
Skip13Year = load('Juv_DSkipping13YearsPondDraining.dat');
Skip14Year = load('Juv_DSkipping14YearsPondDraining.dat');
Skip15Year = load('Juv_DSkipping15YearsPondDraining.dat');


% Organization.
for n = 101:200
    Pond2(n-100,1)=Skip0Year(n,2);
    Pond2(n-100,2)=Skip1Year(n,2);
    Pond2(n-100,3)=Skip2Year(n,2);
```

```
Pond2(n-100,4)=Skip3Year(n,2);
Pond2(n-100,5)=Skip4Year(n,2);
Pond2(n-100,6)=Skip5Year(n,2);
Pond2(n-100,7)=Skip6Year(n,2);
Pond2(n-100,8)=Skip7Year(n,2);
Pond2(n-100,9)=Skip8Year(n,2);
Pond2(n-100,10)=Skip9Year(n,2);
Pond2(n-100,11)=Skip10Year(n,2);
Pond2(n-100,12)=Skip11Year(n,2);
Pond2(n-100,13)=Skip12Year(n,2);
Pond2(n-100,14)=Skip13Year(n,2);
Pond2(n-100,15)=Skip14Year(n,2);
Pond2(n-100,16)=Skip15Year(n,2);

Pond3(n-100,1)=Skip0Year(n,3);
Pond3(n-100,2)=Skip1Year(n,3);
Pond3(n-100,3)=Skip2Year(n,3);
Pond3(n-100,4)=Skip3Year(n,3);
Pond3(n-100,5)=Skip4Year(n,3);
Pond3(n-100,6)=Skip5Year(n,3);
Pond3(n-100,7)=Skip6Year(n,3);
Pond3(n-100,8)=Skip7Year(n,3);
Pond3(n-100,9)=Skip8Year(n,3);
Pond3(n-100,10)=Skip9Year(n,3);
Pond3(n-100,11)=Skip10Year(n,3);
Pond3(n-100,12)=Skip11Year(n,3);
Pond3(n-100,13)=Skip12Year(n,3);
Pond3(n-100,14)=Skip13Year(n,3);
Pond3(n-100,15)=Skip14Year(n,3);
Pond3(n-100,16)=Skip15Year(n,3);

Pond4(n-100,1)=Skip0Year(n,4);
Pond4(n-100,2)=Skip1Year(n,4);
Pond4(n-100,3)=Skip2Year(n,4);
Pond4(n-100,4)=Skip3Year(n,4);
Pond4(n-100,5)=Skip4Year(n,4);
Pond4(n-100,6)=Skip5Year(n,4);
```

```
Pond4(n-100,7)=Skip6Year(n,4);
Pond4(n-100,8)=Skip7Year(n,4);
Pond4(n-100,9)=Skip8Year(n,4);
Pond4(n-100,10)=Skip9Year(n,4);
Pond4(n-100,11)=Skip10Year(n,4);
Pond4(n-100,12)=Skip11Year(n,4);
Pond4(n-100,13)=Skip12Year(n,4);
Pond4(n-100,14)=Skip13Year(n,4);
Pond4(n-100,15)=Skip14Year(n,4);
Pond4(n-100,16)=Skip15Year(n,4);

Pond5(n-100,1)=Skip0Year(n,5);
Pond5(n-100,2)=Skip1Year(n,5);
Pond5(n-100,3)=Skip2Year(n,5);
Pond5(n-100,4)=Skip3Year(n,5);
Pond5(n-100,5)=Skip4Year(n,5);
Pond5(n-100,6)=Skip5Year(n,5);
Pond5(n-100,7)=Skip6Year(n,5);
Pond5(n-100,8)=Skip7Year(n,5);
Pond5(n-100,9)=Skip8Year(n,5);
Pond5(n-100,10)=Skip9Year(n,5);
Pond5(n-100,11)=Skip10Year(n,5);
Pond5(n-100,12)=Skip11Year(n,5);
Pond5(n-100,13)=Skip12Year(n,5);
Pond5(n-100,14)=Skip13Year(n,5);
Pond5(n-100,15)=Skip14Year(n,5);
Pond5(n-100,16)=Skip15Year(n,5);

Pond6(n-100,1)=Skip0Year(n,6);
Pond6(n-100,2)=Skip1Year(n,6);
Pond6(n-100,3)=Skip2Year(n,6);
Pond6(n-100,4)=Skip3Year(n,6);
Pond6(n-100,5)=Skip4Year(n,6);
Pond6(n-100,6)=Skip5Year(n,6);
Pond6(n-100,7)=Skip6Year(n,6);
Pond6(n-100,8)=Skip7Year(n,6);
Pond6(n-100,9)=Skip8Year(n,6);
```

```
    Pond6(n-100,10)=Skip9Year(n,6);
    Pond6(n-100,11)=Skip10Year(n,6);
    Pond6(n-100,12)=Skip11Year(n,6);
    Pond6(n-100,13)=Skip12Year(n,6);
    Pond6(n-100,14)=Skip13Year(n,6);
    Pond6(n-100,15)=Skip14Year(n,6);
    Pond6(n-100,16)=Skip15Year(n,6);

    Pond8(n-100,1)=Skip0Year(n,8);
    Pond8(n-100,2)=Skip1Year(n,8);
    Pond8(n-100,3)=Skip2Year(n,8);
    Pond8(n-100,4)=Skip3Year(n,8);
    Pond8(n-100,5)=Skip4Year(n,8);
    Pond8(n-100,6)=Skip5Year(n,8);
    Pond8(n-100,7)=Skip6Year(n,8);
    Pond8(n-100,8)=Skip7Year(n,8);
    Pond8(n-100,9)=Skip8Year(n,8);
    Pond8(n-100,10)=Skip9Year(n,8);
    Pond8(n-100,11)=Skip10Year(n,8);
    Pond8(n-100,12)=Skip11Year(n,8);
    Pond8(n-100,13)=Skip12Year(n,8);
    Pond8(n-100,14)=Skip13Year(n,8);
    Pond8(n-100,15)=Skip14Year(n,8);
    Pond8(n-100,16)=Skip15Year(n,8);


end

% Mean and SD calculations.
y2 = mean(Pond2,1);
e2 = std(Pond2,1,1);
y3 = mean(Pond3,1);
e3 = std(Pond3,1,1);
y4 = mean(Pond4,1);
e4 = std(Pond4,1,1);
y5 = mean(Pond5,1);
e5 = std(Pond5,1,1);
```

```
y6 = mean(Pond6,1);
e6 = std(Pond6,1,1);
y8 = mean(Pond8,1);
e8 = std(Pond8,1,1);

t = 0:1:15;

% Here's the figure!
figure(1)
hold on
errorbar(t,y2,e2,'og');
errorbar(t,y3,e3,'xc');
errorbar(t,y4,e4,'xk');
errorbar(t,y5,e5,'xm');
errorbar(t,y6,e6,'or');
errorbar(t,y8,e8,'xb');

title('Skipping Years: Dip Netting');
xlabel('Number of Years Skipped Between Drainings');
ylabel('Average CRLF Juvenile Population');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% In this simulation, we assume ponds have drift fences around them every
% year, then every other year, then every two years and so on until there
% are 15 years between management applications.  Each simulation runs for
% 200 years and info is stored in order to create a figure using
% DriftFenceSkipFigure.m

close all;
clear;

% Number of ponds.

nponds = 8;
```

```
% Pond distance matrix initialization.  Called M in thesis.

Dnbyn = zeros(nponds,nponds);

% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);

Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
```

```
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;

% Max distance for bullfrogs.
for i = 1:nponds
```

```
    for j = 1:nponds
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
    end
end

% Max distance for CRLFs.
for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
end



% Initial conditions.
u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.

% This IC can be used to initialize a single bullfrog.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0];

% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u';
catesbeiana(:,1) = v';

% Creating zero matrices.
```

```
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;

% Parameters.
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

% f-1 is the number of years skipped.
for f = 1:1:16
    for i = 1:nsteps
        for j = 1:nponds
        % Assigning the constant entries.
            D(j*3-2,3*j) = r*p4;
            D(3*j,3*j) = p4;
            C(j*5-1,j*5-2) = s3;
            C(j*5,j*5) = s5;

        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.
            if j == 1
                C(j*5-3,j*5-4) = 0;
            elseif j == 2
                C(j*5-3,j*5-4) = 0;
            elseif j == 6
                C(j*5-3,j*5-4) = 0;
            elseif j == 7
                C(j*5-3,j*5-4) = 0;
```

```
            else
                C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
            end
            C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
        end

    % Assigning the function entries.
        for j = 1:nponds
            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5,j*5-1) = s4;

            % Here we are implementing the skipping of years.
        if rem(i,f)==0
            C(j*5-2,j*5-3) = 0;
            C(j*5-2,j*5-4) = 0;
        end

        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.
            Juv_D(j,1)=draytonii(j*3-1,i);
            Juv_C1(j,1)=catesbeiana(j*5-2,i);
            Juv_C2(j,1)=catesbeiana(j*5-1,i);

       % This bit does the CRLF habibtat quality indicator.
            Ad_C(j,1) = catesbeiana(j*5,i);
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
            if Ad_C(j)<1
                Ad_C(j)=1;
            else
                Ad_C(j)=1-(1/(Ad_C(j)));
            end
```

```
            if Ad_C(j)==0
                Ad_C(j)=1;
            end
        end

    % Here we create our gamma distributions which tells us what proportion
    % of the juvenile population will move to which pond.  Note that since
    % this is the 'stochastic' version of the simulation, we update the
    % gamma distribution every year (time step) for each pond.
    % Furthermore, each updated gamma distribution is dependent upon the
    % source pond's juvenile population.  Distance values are drawn and
    % used in their corresponding gamma distribution to obtain rates of
    % dispersion (immigration out of a pond).
        [R_D]=calculate_rij_stoch_Dnew(DnbynD,Juv_D);
        [R_C1]=calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
        [R_C2]=calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

% Stage specific choice movement probabilites.
        newDnbynC2 = zeros(nponds,nponds);
        RecipnewDnbynC2 = zeros(nponds,nponds);
        [rowC2,colC2] = find(R_C2);
        DispDepInflC2 = zeros(nponds,nponds);
        for g = 1:length(rowC2)
                newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
        end

        for h = 1:nponds

            if nnz(R_D(h,:)) > 1
                R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
            end
            if nnz(R_C1(h,:)) > 1
                R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
            end

            for k = 1:nponds
                if newDnbynC2(h,k)==0
```

```
            DispDepInflC2(h,k) = 0;
        else
            RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);
        end
    end
end
for j = 1:length(rowC2)
        if colC2(j)==1
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==2
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==6
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==7
            R_C2(rowC2(j),colC2(j))=0;
        end
end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
end
for j = 1:nponds
    for k = 1:nponds
        R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
    end
end
for j = i:nponds
    for k = 1:nponds
        if Ad_C(j)~=0
            R_D(j,k) = R_D(j,k)*Ad_C(j);
        end

    end
end
```

```
    R_Dtrans = R_D';
    R_C1trans = R_C1';
    R_C2trans = R_C2';

         % Immigration vector calculation.

    for h = 1:nponds
        immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
        immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
        immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
    end

    immVectorJuvD = sum(immMatrixJuvD,2);
    immVectorJuvC1 = sum(immMatrixJuvC1,2);
    immVectorJuvC2 = sum(immMatrixJuvC2,2);

    % Emigration vector calclulation.
    emmVectorJuvD = R_Dtrans*Juv_D;
    emmVectorJuvC1 = R_C1trans*Juv_C1;
    emmVectorJuvC2 = R_C2trans*Juv_C2;

% The following three paragraphs are used mearly to put all the above
% immigration/emigration information in vectors of the appropriate size
% to be subtracted from and added to the population vector.
    x_JuvD_dummy=[0 0 1];
    imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
    emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);

    x_JuvC1_dummy=[0 0 0 1 0];
    imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
    emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);

    x_JuvC2_dummy=[0 0 0 0 1];
    imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
    emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

% Here we update the population vector according to it's parameters and
```

```
    % the immigration/emigration rules outlined above.
        draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
        catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')
+(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
        neg_d=find(draytonii<0); draytonii(neg_d)=0;
        neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;

%  This saves info for each simulation run.
        fid = fopen(['Juv_DSkipping' num2str(f-1) 'YearsDriftFence.dat' ],'a');
 %fprintf(fid, '%3.6f ', i);
        fprintf(fid, '%3.6f ', Juv_D);fprintf(fid,'\n');
        fclose(fid);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code makes a figure using data collected by running
% SkipDriftFence.m

% Calling files.
Skip0Year = load('Juv_DSkipping0YearsDriftFence.dat');
Skip1Year = load('Juv_DSkipping1YearsDriftFence.dat');
Skip2Year = load('Juv_DSkipping2YearsDriftFence.dat');
Skip3Year = load('Juv_DSkipping3YearsDriftFence.dat');
Skip4Year = load('Juv_DSkipping4YearsDriftFence.dat');
Skip5Year = load('Juv_DSkipping5YearsDriftFence.dat');
Skip6Year = load('Juv_DSkipping6YearsDriftFence.dat');
Skip7Year = load('Juv_DSkipping7YearsDriftFence.dat');
Skip8Year = load('Juv_DSkipping8YearsDriftFence.dat');
Skip9Year = load('Juv_DSkipping9YearsDriftFence.dat');
Skip10Year = load('Juv_DSkipping10YearsDriftFence.dat');
Skip11Year = load('Juv_DSkipping11YearsDriftFence.dat');
Skip12Year = load('Juv_DSkipping12YearsDriftFence.dat');
```

```
Skip13Year = load('Juv_DSkipping13YearsDriftFence.dat');
Skip14Year = load('Juv_DSkipping14YearsDriftFence.dat');
Skip15Year = load('Juv_DSkipping15YearsDriftFence.dat');


% Organization.
for n = 101:200
    Pond2(n-100,1)=Skip0Year(n,2);
    Pond2(n-100,2)=Skip1Year(n,2);
    Pond2(n-100,3)=Skip2Year(n,2);
    Pond2(n-100,4)=Skip3Year(n,2);
    Pond2(n-100,5)=Skip4Year(n,2);
    Pond2(n-100,6)=Skip5Year(n,2);
    Pond2(n-100,7)=Skip6Year(n,2);
    Pond2(n-100,8)=Skip7Year(n,2);
    Pond2(n-100,9)=Skip8Year(n,2);
    Pond2(n-100,10)=Skip9Year(n,2);
    Pond2(n-100,11)=Skip10Year(n,2);
    Pond2(n-100,12)=Skip11Year(n,2);
    Pond2(n-100,13)=Skip12Year(n,2);
    Pond2(n-100,14)=Skip13Year(n,2);
    Pond2(n-100,15)=Skip14Year(n,2);
    Pond2(n-100,16)=Skip15Year(n,2);

    Pond3(n-100,1)=Skip0Year(n,3);
    Pond3(n-100,2)=Skip1Year(n,3);
    Pond3(n-100,3)=Skip2Year(n,3);
    Pond3(n-100,4)=Skip3Year(n,3);
    Pond3(n-100,5)=Skip4Year(n,3);
    Pond3(n-100,6)=Skip5Year(n,3);
    Pond3(n-100,7)=Skip6Year(n,3);
    Pond3(n-100,8)=Skip7Year(n,3);
    Pond3(n-100,9)=Skip8Year(n,3);
    Pond3(n-100,10)=Skip9Year(n,3);
    Pond3(n-100,11)=Skip10Year(n,3);
    Pond3(n-100,12)=Skip11Year(n,3);
    Pond3(n-100,13)=Skip12Year(n,3);
```

```
Pond3(n-100,14)=Skip13Year(n,3);
Pond3(n-100,15)=Skip14Year(n,3);
Pond3(n-100,16)=Skip15Year(n,3);

Pond4(n-100,1)=Skip0Year(n,4);
Pond4(n-100,2)=Skip1Year(n,4);
Pond4(n-100,3)=Skip2Year(n,4);
Pond4(n-100,4)=Skip3Year(n,4);
Pond4(n-100,5)=Skip4Year(n,4);
Pond4(n-100,6)=Skip5Year(n,4);
Pond4(n-100,7)=Skip6Year(n,4);
Pond4(n-100,8)=Skip7Year(n,4);
Pond4(n-100,9)=Skip8Year(n,4);
Pond4(n-100,10)=Skip9Year(n,4);
Pond4(n-100,11)=Skip10Year(n,4);
Pond4(n-100,12)=Skip11Year(n,4);
Pond4(n-100,13)=Skip12Year(n,4);
Pond4(n-100,14)=Skip13Year(n,4);
Pond4(n-100,15)=Skip14Year(n,4);
Pond4(n-100,16)=Skip15Year(n,4);

Pond5(n-100,1)=Skip0Year(n,5);
Pond5(n-100,2)=Skip1Year(n,5);
Pond5(n-100,3)=Skip2Year(n,5);
Pond5(n-100,4)=Skip3Year(n,5);
Pond5(n-100,5)=Skip4Year(n,5);
Pond5(n-100,6)=Skip5Year(n,5);
Pond5(n-100,7)=Skip6Year(n,5);
Pond5(n-100,8)=Skip7Year(n,5);
Pond5(n-100,9)=Skip8Year(n,5);
Pond5(n-100,10)=Skip9Year(n,5);
Pond5(n-100,11)=Skip10Year(n,5);
Pond5(n-100,12)=Skip11Year(n,5);
Pond5(n-100,13)=Skip12Year(n,5);
Pond5(n-100,14)=Skip13Year(n,5);
Pond5(n-100,15)=Skip14Year(n,5);
Pond5(n-100,16)=Skip15Year(n,5);
```

```
    Pond6(n-100,1)=Skip0Year(n,6);
    Pond6(n-100,2)=Skip1Year(n,6);
    Pond6(n-100,3)=Skip2Year(n,6);
    Pond6(n-100,4)=Skip3Year(n,6);
    Pond6(n-100,5)=Skip4Year(n,6);
    Pond6(n-100,6)=Skip5Year(n,6);
    Pond6(n-100,7)=Skip6Year(n,6);
    Pond6(n-100,8)=Skip7Year(n,6);
    Pond6(n-100,9)=Skip8Year(n,6);
    Pond6(n-100,10)=Skip9Year(n,6);
    Pond6(n-100,11)=Skip10Year(n,6);
    Pond6(n-100,12)=Skip11Year(n,6);
    Pond6(n-100,13)=Skip12Year(n,6);
    Pond6(n-100,14)=Skip13Year(n,6);
    Pond6(n-100,15)=Skip14Year(n,6);
    Pond6(n-100,16)=Skip15Year(n,6);

    Pond8(n-100,1)=Skip0Year(n,8);
    Pond8(n-100,2)=Skip1Year(n,8);
    Pond8(n-100,3)=Skip2Year(n,8);
    Pond8(n-100,4)=Skip3Year(n,8);
    Pond8(n-100,5)=Skip4Year(n,8);
    Pond8(n-100,6)=Skip5Year(n,8);
    Pond8(n-100,7)=Skip6Year(n,8);
    Pond8(n-100,8)=Skip7Year(n,8);
    Pond8(n-100,9)=Skip8Year(n,8);
    Pond8(n-100,10)=Skip9Year(n,8);
    Pond8(n-100,11)=Skip10Year(n,8);
    Pond8(n-100,12)=Skip11Year(n,8);
    Pond8(n-100,13)=Skip12Year(n,8);
    Pond8(n-100,14)=Skip13Year(n,8);
    Pond8(n-100,15)=Skip14Year(n,8);
    Pond8(n-100,16)=Skip15Year(n,8);


end
```

```
% Calculating means and SD's.
y2 = mean(Pond2,1);
e2 = std(Pond2,1,1);
y3 = mean(Pond3,1);
e3 = std(Pond3,1,1);
y4 = mean(Pond4,1);
e4 = std(Pond4,1,1);
y5 = mean(Pond5,1);
e5 = std(Pond5,1,1);
y6 = mean(Pond6,1);
e6 = std(Pond6,1,1);
y8 = mean(Pond8,1);
e8 = std(Pond8,1,1);

t = 0:1:15;


% The figure!
figure(1)
hold on
errorbar(t,y2,e2,'og');
errorbar(t,y3,e3,'xc');
errorbar(t,y4,e4,'xk');
errorbar(t,y5,e5,'xm');
errorbar(t,y6,e6,'or');
errorbar(t,y8,e8,'xb');

title('Skipping Years: Drift Fence');
xlabel('Number of Years Skipped Between Management Applications');
ylabel('Average CRLF Juvenile Population');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% In this simulation, we assume terrestrial bullfrogs are shot every
% year, then every other year, then every two years and so on until there
```

```
% are 15 years between management applications.  Each simulation runs for
% 200 years and info is stored in order to create a figure using
% ShootingSkipFigure.m

close all;
clear;

% Number of ponds.

nponds = 8;

% Pond distance matrix initialization.  Called M in thesis.

Dnbyn = zeros(nponds,nponds);

% Pond distance matrix entry values.

Dnbyn(1,2) =  14740;
Dnbyn(2,1) = Dnbyn(1,2);
Dnbyn(1,3) =  15240;
Dnbyn(3,1) = Dnbyn(1,3);
Dnbyn(1,4) =  15640;
Dnbyn(4,1) = Dnbyn(1,4);
Dnbyn(1,5) =  21610;
Dnbyn(5,1) = Dnbyn(1,5);
Dnbyn(1,6) =  22400;
Dnbyn(6,1) = Dnbyn(1,6);
Dnbyn(1,7) =  30830;
Dnbyn(7,1) = Dnbyn(1,7);
Dnbyn(1,8) =  43870;
Dnbyn(8,1) = Dnbyn(1,8);

Dnbyn(2,3) =  519.38;
Dnbyn(3,2) = Dnbyn(2,3);
Dnbyn(2,4) =  885.14;
Dnbyn(4,2) = Dnbyn(2,4);
Dnbyn(2,5) =  6950;
```

```
Dnbyn(5,2) = Dnbyn(2,5);
Dnbyn(2,6) =  7720;
Dnbyn(6,2) = Dnbyn(2,6);
Dnbyn(2,7) =  16130;
Dnbyn(7,2) = Dnbyn(2,7);
Dnbyn(2,8) =  29130;
Dnbyn(8,2) = Dnbyn(2,8);

Dnbyn(3,4) =  541.12;
Dnbyn(4,3) = Dnbyn(3,4);
Dnbyn(3,5) =  6440;
Dnbyn(5,3) = Dnbyn(3,5);
Dnbyn(3,6) =  7210;
Dnbyn(6,3) = Dnbyn(3,6);
Dnbyn(3,7) =  15610;
Dnbyn(7,3) = Dnbyn(3,7);
Dnbyn(3,8) =  28650;
Dnbyn(8,3) = Dnbyn(3,8);

Dnbyn(4,5) =  6040;
Dnbyn(5,4) = Dnbyn(4,5);
Dnbyn(4,6) =  6840;
Dnbyn(6,4) = Dnbyn(4,6);
Dnbyn(4,7) =  15210;
Dnbyn(7,4) = Dnbyn(4,7);
Dnbyn(4,8) =  28210;
Dnbyn(8,4) = Dnbyn(4,8);

Dnbyn(5,6) =  677.97;
Dnbyn(6,5) = Dnbyn(5,6);
Dnbyn(5,7) =  9110;
Dnbyn(7,5) = Dnbyn(5,7);
Dnbyn(5,8) =  22270;
Dnbyn(8,5) = Dnbyn(5,8);

Dnbyn(6,7) =  8400;
Dnbyn(7,6) = Dnbyn(6,7);
```

```
Dnbyn(6,8) =  21580;
Dnbyn(8,6) = Dnbyn(6,8);

Dnbyn(7,8) =  13340;
Dnbyn(8,7) = Dnbyn(7,8);

DnbynD = Dnbyn;
DnbynC = Dnbyn;

% Bullfrog max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynC(i,j) > 5600
            DnbynC(i,j) = 0;
        end
    end
end

% CRLF max distance.
for i = 1:nponds
    for j = 1:nponds
        if DnbynD(i,j) > 2800
            DnbynD(i,j) = 0;
        end
    end
end


% Initial conditions.
u = [4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4 4956 42 4
 4956 42 4 ];  % Initial CRLF population densities.
v = [0 0 0 0 0 0 0 0 0 0 9158 754 68 16 4 9158 754 68 16 4 9158 754 68 16
4 0 0 0 0 0 0 0 0 0 9158 754 68 16 4];  % Initial BF population densities.
%v = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0];
```

```
% Here we initalize a zero vector of the appropriate size that is of a form
% that is easier to work with later on.
draytonii = zeros(nponds*3,1);
catesbeiana = zeros(nponds*5,1);

% Now we throw in the inital conditions.
draytonii(:,1) = u';
catesbeiana(:,1) = v';

% Creating zero matrices.
D = zeros(nponds*3,nponds*3);
C = zeros(nponds*5,nponds*5);

% Number of timesteps in the simulation.
nsteps = 200;

% Parameters.
p1 = 0.025; p2 = 0.25; p3 = 0.4; p4 = 0.5; r = 1500;
s1 = 0.1; s2 = 0.02; sFT = 0.016; s3 = 0.26; s4 = 0.32;
s5 = 0.65; b = 4000; gamma = 0.02; mu = 0.05; eta = 0.033;

% Unknown parameters.
%alphaM1 = 0.01; alphaM2 = 0.002; alphaM0 = 0.00002;%200year coexistence
%alphaM1 = 0.01; alphaM2 = 0.003; alphaM0 = 0.00002;%100year coexistence
alphaM1 = 0.001; alphaM2 = 0.0008; alphaM0 = 0.00003;%60 year coexistence
%alphaM1 = 0.003; alphaM2 = 0.0001; alphaM0 = 0.00004;%20year coexistence

% f-1 is the number of years skipped.
for f = 1:1:16
    for i = 1:nsteps
        for j = 1:nponds
        % Assigning the constant entries.
            D(j*3-2,3*j) = r*p4;
            D(3*j,3*j) = p4;
            C(j*5-1,j*5-2) = s3;
            C(j*5,j*5) = s5;
```

```
        % Here we ensure that overwintering bullfrog tadpoles will not
        % survive in the seasonal ponds.
            if j == 1
                C(j*5-3,j*5-4) = 0;
            elseif j == 2
                C(j*5-3,j*5-4) = 0;
            elseif j == 6
                C(j*5-3,j*5-4) = 0;
            elseif j == 7
                C(j*5-3,j*5-4) = 0;
            else
                C(j*5-3,j*5-4) = s1*exp(-gamma*catesbeiana(5*j,i));
            end
            C(j*5-4,j*5) = b*s5*exp(-gamma*catesbeiana(5*j,i));
        end

    % Assigning the function entries.
        for j = 1:nponds
            D(j*3-1,j*3-2) = p1*p2*exp(-eta*draytonii(3*j,i)-alphaM1
*catesbeiana(5*j,i)-alphaM0*catesbeiana(5*j-4,i));
            D(j*3,j*3-1) = p3*exp(-alphaM2*catesbeiana(5*j,i));
            C(j*5-2,j*5-4) = sFT*exp(-mu*catesbeiana(5*j,i));
            C(j*5-2,j*5-3) = s2*exp(-mu*catesbeiana(5*j,i));
            C(j*5,j*5-1) = s4;
         % This is the part that skips years for us.
        if rem(i,f)==0
            C(j*5-1,j*5-2) = 0;
            C(j*5,j*5-1) = 0;
            C(j*5,j*5) = 0;
        end

        % Here we gather the juvenile populations of both species since
        % they will be the only indiviuals moving between ponds.
            Juv_D(j,1)=draytonii(j*3-1,i);
            Juv_C1(j,1)=catesbeiana(j*5-2,i);
            Juv_C2(j,1)=catesbeiana(j*5-1,i);
```

```
        % CRLF habitat quality indicator.
        Ad_C(j,1) = catesbeiana(j*5,i);
        if Ad_C(j)==0
            Ad_C(j)=1;
        end
        if Ad_C(j)<1
            Ad_C(j)=1;
        else
            Ad_C(j)=1-(1/(Ad_C(j)));
        end
        if Ad_C(j)==0
            Ad_C(j)=1;
        end
    end

% Here we create our gamma distributions which tells us what proportion
% of the juvenile population will move to which pond.  Note that since
% this is the 'stochastic' version of the simulation, we update the
% gamma distribution every year (time step) for each pond.
% Furthermore, each updated gamma distribution is dependent upon the
% source pond's juvenile population.  Distance values are drawn and
% used in their corresponding gamma distribution to obtain rates of
% dispersion (immigration out of a pond).
    [R_D]=calculate_rij_stoch_Dnew(DnbynD,Juv_D);
    [R_C1]=calculate_rij_stoch_Cnew(DnbynC,Juv_C1);
    [R_C2]=calculate_rij_stoch_Cnew(DnbynC,Juv_C2);

    % Stage specific movement choice probability.
    newDnbynC2 = zeros(nponds,nponds);
    RecipnewDnbynC2 = zeros(nponds,nponds);
    [rowC2,colC2] = find(R_C2);
    DispDepInflC2 = zeros(nponds,nponds);
    for g = 1:length(rowC2)
            newDnbynC2(rowC2(g),colC2(g)) = DnbynC(rowC2(g),colC2(g));
    end

    for h = 1:nponds
```

```
    if nnz(R_D(h,:)) > 1
        R_D(h,:) = R_D(h,:)/nnz(R_D(h,:));
    end
    if nnz(R_C1(h,:)) > 1
        R_C1(h,:) = R_C1(h,:)/nnz(R_C1(h,:));
    end
    for k = 1:nponds
        if newDnbynC2(h,k)==0
            DispDepInflC2(h,k) = 0;
        else
            RecipnewDnbynC2(h,k) = 1/newDnbynC2(h,k);
        end
    end
end
for j = 1:length(rowC2)
        if colC2(j)==1
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==2
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==6
            R_C2(rowC2(j),colC2(j))=0;
        elseif colC2(j)==7
            R_C2(rowC2(j),colC2(j))=0;
        end
end
for h = 1:nponds
    for k = 1:nponds
        if RecipnewDnbynC2(h,k)~=0
            DispDepInflC2(h,k) = (RecipnewDnbynC2(h,k))/(sum(RecipnewDnbynC2(h,:)));
        end
    end
end
for j = 1:nponds
    for k = 1:nponds
        R_C2(j,k) = R_C2(j,k)*DispDepInflC2(j,k);
    end
```

```
    end
    for j = i:nponds
        for k = 1:nponds
            if Ad_C(j)~=0
                R_D(j,k) = R_D(j,k)*Ad_C(j);
            end
        end
    end
    R_Dtrans = R_D';
    R_C1trans = R_C1';
    R_C2trans = R_C2';

    % Immigration vector calculation.
    for h = 1:nponds
        immMatrixJuvD(h,:) = R_D(h,:)*Juv_D(h);
        immMatrixJuvC1(h,:) = R_C1(h,:)*Juv_C1(h);
        immMatrixJuvC2(h,:) = R_C2(h,:)*Juv_C2(h);
    end

    immVectorJuvD = sum(immMatrixJuvD,2);
    immVectorJuvC1 = sum(immMatrixJuvC1,2);
    immVectorJuvC2 = sum(immMatrixJuvC2,2);

    % Emigration vector calculation.
    emmVectorJuvD = R_Dtrans*Juv_D;
    emmVectorJuvC1 = R_C1trans*Juv_C1;
    emmVectorJuvC2 = R_C2trans*Juv_C2;

% The following three paragraphs are used mearly to put all the above
% immigration/emigration information in vectors of the appropriate size
% to be subtracted from and added to the population vector.
    x_JuvD_dummy=[0 0 1];
    imm_JuvD=kron(immVectorJuvD',x_JuvD_dummy);
    emm_JuvD=kron(emmVectorJuvD',x_JuvD_dummy);


    x_JuvC1_dummy=[0 0 0 1 0];
```

```
        imm_JuvC1=kron(immVectorJuvC1',x_JuvC1_dummy);
        emm_JuvC1=kron(emmVectorJuvC1',x_JuvC1_dummy);


        x_JuvC2_dummy=[0 0 0 0 1];
        imm_JuvC2=kron(immVectorJuvC2',x_JuvC2_dummy);
        emm_JuvC2=kron(emmVectorJuvC2',x_JuvC2_dummy);

    % Here we update the population vector according to it's parameters and
    % the immigration/emigration rules outlined above.
        draytonii(:,i+1) = D*draytonii(:,i)-imm_JuvD'+emm_JuvD';
        catesbeiana(:,i+1) = C*catesbeiana(:,i)-(imm_JuvC1'+imm_JuvC2')
+(emm_JuvC1'+emm_JuvC2');

    % This ensures that we do not deal with (or see in the figures)
    % negative numbers of frogs.
        neg_d=find(draytonii<0); draytonii(neg_d)=0;
        neg_c=find(catesbeiana<0); catesbeiana(neg_c)=0;

% Here we save data for each simulation run.
        fid = fopen(['Juv_DSkipping' num2str(f-1) 'YearsShooting.dat' ],'a');
 %fprintf(fid, '%3.6f ', i);
        fprintf(fid, '%3.6f ', Juv_D);fprintf(fid,'\n');
        fclose(fid);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code makes a figure using data collected by running
% SkipShooting.m

% Calling files.
Skip0Year = load('Juv_DSkipping0YearsShooting.dat');
Skip1Year = load('Juv_DSkipping1YearsShooting.dat');
Skip2Year = load('Juv_DSkipping2YearsShooting.dat');
Skip3Year = load('Juv_DSkipping3YearsShooting.dat');
```

```
Skip4Year = load('Juv_DSkipping4YearsShooting.dat');
Skip5Year = load('Juv_DSkipping5YearsShooting.dat');
Skip6Year = load('Juv_DSkipping6YearsShooting.dat');
Skip7Year = load('Juv_DSkipping7YearsShooting.dat');
Skip8Year = load('Juv_DSkipping8YearsShooting.dat');
Skip9Year = load('Juv_DSkipping9YearsShooting.dat');
Skip10Year = load('Juv_DSkipping10YearsShooting.dat');
Skip11Year = load('Juv_DSkipping11YearsShooting.dat');
Skip12Year = load('Juv_DSkipping12YearsShooting.dat');
Skip13Year = load('Juv_DSkipping13YearsShooting.dat');
Skip14Year = load('Juv_DSkipping14YearsShooting.dat');
Skip15Year = load('Juv_DSkipping15YearsShooting.dat');


% Organization.
for n = 101:200
    Pond2(n-100,1)=Skip0Year(n,2);
    Pond2(n-100,2)=Skip1Year(n,2);
    Pond2(n-100,3)=Skip2Year(n,2);
    Pond2(n-100,4)=Skip3Year(n,2);
    Pond2(n-100,5)=Skip4Year(n,2);
    Pond2(n-100,6)=Skip5Year(n,2);
    Pond2(n-100,7)=Skip6Year(n,2);
    Pond2(n-100,8)=Skip7Year(n,2);
    Pond2(n-100,9)=Skip8Year(n,2);
    Pond2(n-100,10)=Skip9Year(n,2);
    Pond2(n-100,11)=Skip10Year(n,2);
    Pond2(n-100,12)=Skip11Year(n,2);
    Pond2(n-100,13)=Skip12Year(n,2);
    Pond2(n-100,14)=Skip13Year(n,2);
    Pond2(n-100,15)=Skip14Year(n,2);
    Pond2(n-100,16)=Skip15Year(n,2);

    Pond3(n-100,1)=Skip0Year(n,3);
    Pond3(n-100,2)=Skip1Year(n,3);
    Pond3(n-100,3)=Skip2Year(n,3);
    Pond3(n-100,4)=Skip3Year(n,3);
```

```
Pond3(n-100,5)=Skip4Year(n,3);
Pond3(n-100,6)=Skip5Year(n,3);
Pond3(n-100,7)=Skip6Year(n,3);
Pond3(n-100,8)=Skip7Year(n,3);
Pond3(n-100,9)=Skip8Year(n,3);
Pond3(n-100,10)=Skip9Year(n,3);
Pond3(n-100,11)=Skip10Year(n,3);
Pond3(n-100,12)=Skip11Year(n,3);
Pond3(n-100,13)=Skip12Year(n,3);
Pond3(n-100,14)=Skip13Year(n,3);
Pond3(n-100,15)=Skip14Year(n,3);
Pond3(n-100,16)=Skip15Year(n,3);

Pond4(n-100,1)=Skip0Year(n,4);
Pond4(n-100,2)=Skip1Year(n,4);
Pond4(n-100,3)=Skip2Year(n,4);
Pond4(n-100,4)=Skip3Year(n,4);
Pond4(n-100,5)=Skip4Year(n,4);
Pond4(n-100,6)=Skip5Year(n,4);
Pond4(n-100,7)=Skip6Year(n,4);
Pond4(n-100,8)=Skip7Year(n,4);
Pond4(n-100,9)=Skip8Year(n,4);
Pond4(n-100,10)=Skip9Year(n,4);
Pond4(n-100,11)=Skip10Year(n,4);
Pond4(n-100,12)=Skip11Year(n,4);
Pond4(n-100,13)=Skip12Year(n,4);
Pond4(n-100,14)=Skip13Year(n,4);
Pond4(n-100,15)=Skip14Year(n,4);
Pond4(n-100,16)=Skip15Year(n,4);

Pond5(n-100,1)=Skip0Year(n,5);
Pond5(n-100,2)=Skip1Year(n,5);
Pond5(n-100,3)=Skip2Year(n,5);
Pond5(n-100,4)=Skip3Year(n,5);
Pond5(n-100,5)=Skip4Year(n,5);
Pond5(n-100,6)=Skip5Year(n,5);
Pond5(n-100,7)=Skip6Year(n,5);
```

```
Pond5(n-100,8)=Skip7Year(n,5);
Pond5(n-100,9)=Skip8Year(n,5);
Pond5(n-100,10)=Skip9Year(n,5);
Pond5(n-100,11)=Skip10Year(n,5);
Pond5(n-100,12)=Skip11Year(n,5);
Pond5(n-100,13)=Skip12Year(n,5);
Pond5(n-100,14)=Skip13Year(n,5);
Pond5(n-100,15)=Skip14Year(n,5);
Pond5(n-100,16)=Skip15Year(n,5);

Pond6(n-100,1)=Skip0Year(n,6);
Pond6(n-100,2)=Skip1Year(n,6);
Pond6(n-100,3)=Skip2Year(n,6);
Pond6(n-100,4)=Skip3Year(n,6);
Pond6(n-100,5)=Skip4Year(n,6);
Pond6(n-100,6)=Skip5Year(n,6);
Pond6(n-100,7)=Skip6Year(n,6);
Pond6(n-100,8)=Skip7Year(n,6);
Pond6(n-100,9)=Skip8Year(n,6);
Pond6(n-100,10)=Skip9Year(n,6);
Pond6(n-100,11)=Skip10Year(n,6);
Pond6(n-100,12)=Skip11Year(n,6);
Pond6(n-100,13)=Skip12Year(n,6);
Pond6(n-100,14)=Skip13Year(n,6);
Pond6(n-100,15)=Skip14Year(n,6);
Pond6(n-100,16)=Skip15Year(n,6);

Pond8(n-100,1)=Skip0Year(n,8);
Pond8(n-100,2)=Skip1Year(n,8);
Pond8(n-100,3)=Skip2Year(n,8);
Pond8(n-100,4)=Skip3Year(n,8);
Pond8(n-100,5)=Skip4Year(n,8);
Pond8(n-100,6)=Skip5Year(n,8);
Pond8(n-100,7)=Skip6Year(n,8);
Pond8(n-100,8)=Skip7Year(n,8);
Pond8(n-100,9)=Skip8Year(n,8);
Pond8(n-100,10)=Skip9Year(n,8);
```

```
    Pond8(n-100,11)=Skip10Year(n,8);
    Pond8(n-100,12)=Skip11Year(n,8);
    Pond8(n-100,13)=Skip12Year(n,8);
    Pond8(n-100,14)=Skip13Year(n,8);
    Pond8(n-100,15)=Skip14Year(n,8);
    Pond8(n-100,16)=Skip15Year(n,8);


end

% Calculating means and SD's.
y2 = mean(Pond2,1);
e2 = std(Pond2,1,1);
y3 = mean(Pond3,1);
e3 = std(Pond3,1,1);
y4 = mean(Pond4,1);
e4 = std(Pond4,1,1);
y5 = mean(Pond5,1);
e5 = std(Pond5,1,1);
y6 = mean(Pond6,1);
e6 = std(Pond6,1,1);
y8 = mean(Pond8,1);
e8 = std(Pond8,1,1);

t = 0:1:15;

% Here's our figure!

figure(1)
hold on
errorbar(t,y2,e2,'og');
errorbar(t,y3,e3,'xc');
errorbar(t,y4,e4,'xk');
errorbar(t,y5,e5,'xm');
errorbar(t,y6,e6,'or');
errorbar(t,y8,e8,'xb');
```

```
title('Skipping Years: Shooting');
xlabel('Number of Years Skipped Between Management Applications');
ylabel('Average CRLF Juvenile Population');
legend('Pond AD', 'Pond WD', 'Pond CP', 'Pond OT', 'Pond MP', 'Pond BL');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```