

Computing Traveling-Wave Front Solutions in a Diffusive Predator-Prey Model

Brian Kim (bgkim@umich.edu)
Professor Gregory Lyng, Advisor
University of Michigan
Department of Mathematics
REU Program
June 23, 2004

Abstract

In this paper we will describe the process behind computing traveling-wave solutions in a diffusive predator-prey system. There is information on the existence of traveling-wave solutions using an abstract topological argument by Dunbar [D] and there are methods for computing traveling-wave solutions by Doedel & Friedman [DF]. But the difference here is that we shall describe how to actually compute these solutions so that they can be viewed and further examined. The main purpose of this paper is to be less theoretical and more practical. If one would like to view the theory behind the methods used here to find the traveling-wave solutions, read Beyn [B].

1 Introduction

Mathematical modeling will always be an important field of mathematics because of its applications to the real world. While no model is perfect, if a close enough approximation can be obtained, then scientists can see how certain factors will affect a situation by merely working out equations on a piece of paper as opposed to actually running an experiment.

One mathematical model that is frequently examined is the Lotka-Volterra predator-prey model. This refers to a system in which there are two populations known as the predator and the prey. The model states that the prey will grow at a certain rate but will also be eaten at a certain rate because of predators. The predators will die at a certain rate but will then grow by eating prey.

In this paper we begin by describing the Lotka-Volterra model in its simplest form, then describe some of the modifications to this model. The first of these modifications is the logistic growth term. This term shall reject the initial model's notion that if there are no predators to eat any of the prey, the prey population will grow infinitely large. This is an unrealistic scenario. Second we consider the effect of spatially distributing the predators and the prey. This step is important because it leads to a partial differential equation (which is necessary for finding traveling waves). Finally, for the last model we explain how to compute the traveling-wave solutions.

A traveling-wave solution can be explained through an example. If there exists a population of plants (prey) spread along a coastline, adding some turtles (predators) at one end would cause a wave of transition to a stable

state. That is, initially there would be a small amount of turtles at the one end, but as they grow in numbers from eating the plants, eventually they will spread out. If one were looking at this process on a space - population graph that changed over time, it would look as if a wave of turtles was moving across the coastline.

Scientists that have studied real pred-prey systems for certain species such as plankton and also in reaction-diffusion equations from neurophysiology, chemistry and epidemiology have seen that traveling waves indeed exist in their respective areas of study. Since traveling waves exist in practice, it is essential that they are represented in the mathematical model. Traveling waves are a special, fundamental class of solutions of PDEs used to represent the transportation of information in a single direction. The ascertainment of the traveling waves was a major discovery in the pred-prey model [M] because of how it represented a transition between equilibria.

2 Construction of the Predator-Prey Model

This initial section is dedicated towards seeing the evolution of the pred-prey model. This will lead to the understanding of each part of the model and give a better feel of what the model is exactly trying to represent. Finally the traveling waves will be inserted into the model, and that will be the starting point for calculating the traveling-wave solutions.

2.1 Lotka-Volterra Pred-Prey Model

The first model was introduced in the 1920s by the American biophysicist Alfred Lotka and the Italian mathematician Vito Volterra [M]. With u representing prey population and w representing predator population

$$\frac{du}{dt} = au - buw, \tag{1}$$

$$\frac{dw}{dt} = -ew + fuw, \tag{2}$$

where a, b, e, f are positive constants. As is apparent from looking at the model, the prey shall grow exponentially depending on the difference between rate of growth, a , and the amount of prey being eaten, bu . The predators shall grow exponentially depending on the difference between the amount of prey being eaten, fu , and the rate of predator death without prey, e . We note that if $w = 0$, then u grows exponentially without bound.

As can be seen in figure 1, there are closed orbits surrounding the critical point $(\frac{a}{b}, \frac{e}{f})$ that implies the solutions are periodic in time and that the maxima of each population are shifted from each other. There is a saddle point at $(0, 0)$ representing the absence of both populations. The saddle point shows what is obvious in real life, if there are prey and no predators, the prey will grow. If there are predators and no prey to eat, then they will die out because of the absence of food.

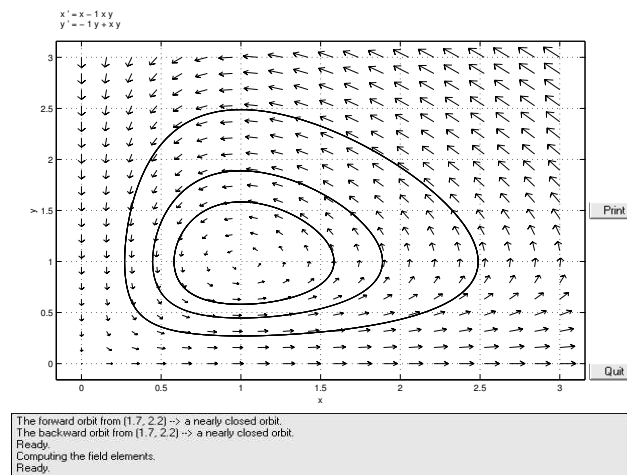


Figure 1: phase plane for prey(x) vs pred(y) in original model

2.2 Logistic Growth

One of the problems with the initially proposed Lotka-Volterra model (1)–(2) is the fact that in the absence of predators, the prey will grow without bound. This is not the case as while the prey continues to grow, space and resources will run out eventually, thereby limiting the growth of the prey population. To handle this case, the pred-prey system can be modified to

$$\frac{du}{dt} = au\left(1 - \frac{u}{K}\right) - buw, \quad (3)$$

$$\frac{dw}{dt} = -ew + f uw, \quad (4)$$

where K is the maximum amount of prey that can exist in the model. Therefore if $K = 5$, then if a small amount of prey was added to a system with no predators, the prey would grow to 5 as $t \rightarrow \infty$ as apposed to the prey growing to ∞ as $t \rightarrow \infty$. From looking at (3), if the prey population is greater

than the carrying capacity, the $(1 - \frac{u}{K})$ term will be negative ensuring that the prey population will decrease.

Now by examining figure 2, there is a saddle point at $(K, 0)$, showing that if any solution starts with more than K prey, then they will quickly go back to the $< K$ region. Also, now instead of there being a spiral point, there is a stable point that all solutions approach as $t \rightarrow \infty$.

There are 3 critical points of the system. One is $(0, 0)$ which is the absence of both species. The second critical point, $(K, 0)$, represents the situation where the prey have reached their carrying capacity in the absence of predators. The final critical point is $(\frac{e}{f}, \frac{a}{b}(1 - \frac{e}{fK}))$, which is the coexistence state where both species are nonzero. The first two critical points are unstable saddle points and the coexistence critical point is a stable critical point that is approached as $t \rightarrow \infty$.

$$u(+\infty) = \frac{e}{f}, \quad w(+\infty) = \frac{a}{b}(1 - \frac{e}{fK}).$$

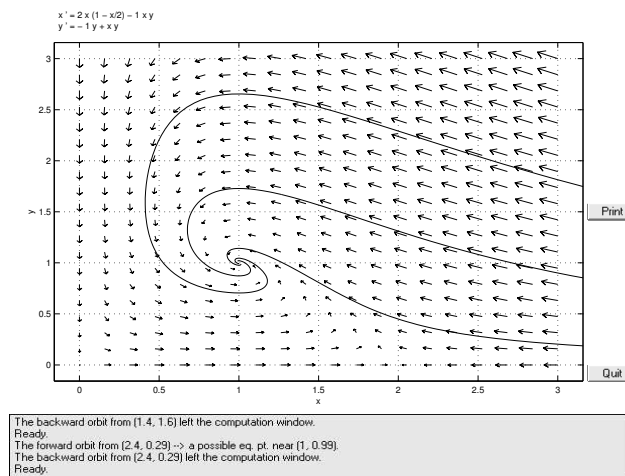


Figure 2: phase plane for prey(x) vs pred(y) with logistic growth

2.3 Reaction Diffusion

An aspect that the previous models neglect is the fact that populations take up some sort of space. Up until now we have been merely analyzing the species' population size, but in the real world the total population will have

some sort of distribution over space (for simplicity we shall analyze one-dimensional space).

Once the concept of space has been added to the equations, another concept must be examined, which is that populations will move between space. For example, if a large amount of some population is crammed into a certain area, they will tend to spread out to maximize productivity. This process is called diffusion.

For the pred-prey model, we shall use a certain approach to diffusion known as Fickian diffusion. This defines the flux of a population, J , as proportional to the gradient of the concentration. In one dimension

$$J = -D \frac{\partial c}{\partial x},$$

where $c(x, t)$ is the concentration of a species and D is its diffusivity (how quickly it will disperse). By examining the classical diffusion equation we find that

$$\begin{aligned} \frac{\partial c}{\partial t} &= -\frac{\partial J}{\partial x}, \\ &= -\frac{\partial(-D \frac{\partial c}{\partial x})}{\partial x}, \\ &= D \frac{\partial^2 c}{\partial x^2}. \end{aligned}$$

Now inserting the diffusion term into the pred-prey equations we have functions for the pred and prey in terms of space and time

$$u_t = D_1 u_{xx} + au(1 - \frac{u}{K}) - buw, \tag{5}$$

$$w_t = D_2 w_{xx} - ew + f uw, \tag{6}$$

where subscripts denote partial differentiation.

2.4 Non Dimensionalization

At this point there is an abundance of parameters. To simplify the model we shall use a technique that makes all parameters *nondimensional*. This has two main advantages. First of all, it reduces the number of parameters that need to be modified when analyzing the equations. Second, it shows

what relationships between the terms are important. By a straightforward calculation we get

$$U_t = DU_{xx} + U(1 - U - W), \quad (7)$$

$$W_t = W_{xx} + \alpha W(U - \beta), \quad (8)$$

where

$$U = \frac{u}{K}, \quad W = \frac{b}{a}w, \quad t' = at, \quad x' = \frac{x}{\sqrt{D_2/a}},$$

$$D = \frac{D_1}{D_2}, \quad \alpha = \frac{fK}{a}, \quad \beta = \frac{e}{fK},$$

and we have dropped the ' on t and x for readability.

One thing to notice is that β is the death rate of predators divided by the maximum rate at which the predators can grow (which is fu as $u \rightarrow K$). Therefore $0 < \beta < 1$ (or else the predators will always die out).

2.5 Traveling waves

In the dimensionless model, the critical points $(K, 0)$ and $(\frac{e}{f}, \frac{a}{b}(1 - \frac{e}{fK}))$ in (u, w) are $(1, 0)$ and $(\beta, 1 - \beta)$ in (U, W) respectively. Since the model is now a PDE, $(1, 0)$ and $(\beta, 1 - \beta)$ are zeros of the reaction part of the PDE.

The next step is to assume that the solutions of the diffusive model (7)—(8) have the shape of a traveling wave as it propagates. While there could be other types of solutions that are not traveling waves, we are only interested in the traveling-wave solutions. Therefore we shall make the assumption that there exists some solutions that have the following form

$$U(t, x) = u(x + ct), \quad W(t, x) = W(x + ct),$$

where c is the wave speed. Now one problem with the model is that it is a PDE which is much more difficult to deal with. To simplify things let $s = x + ct$ (s is called the wave variable). After applying the chain rule, we can get a system in terms of just s . The model becomes

$$cu' = Du'' + u(1 - u - w), \quad (9)$$

$$cw' = w'' + \alpha w(u - \beta), \quad (10)$$

where $' = \frac{d}{ds}$.

The last step is to get a first-order system of equations. This reduces to

$$u' = v, \tag{11}$$

$$v' = \frac{c}{D}v - \frac{1}{D}u(1 - u - w), \tag{12}$$

$$w' = z, \tag{13}$$

$$z' = cz - \alpha w(u - \beta). \tag{14}$$

Now there is a first-order system of equations to try and find a traveling-wave solution for. It should satisfy the following boundary conditions:

$$u(-\infty) = 1, \quad w(-\infty) = 0,$$

$$u(+\infty) = \beta, \quad w(+\infty) = 1 - \beta.$$

Finding a traveling-wave solution of the PDE (7)–(8) is equivalent to finding a trajectory in the four-dimensional phase space that satisfies the above boundary conditions.

3 Traveling-Wave Front Solutions

3.1 Eigenvalues and Eigenvectors

The first step in finding out information about the traveling-wave solutions in the pred-prey model is to analyze the eigenvalues and eigenvectors of the system of equations (11)–(14) at the two critical points. The first thing that this does is it lets us see the local behavior near the critical points. Also from analyzing the eigenvalues and eigenvectors, we can find out how changing the parameters to the pred-prey model will change the behavior of the traveling-wave solutions. For example, changing the value of α will determine whether the critical point, $(1, 0)$, will be approached monotonely in the traveling-wave solution. Finally, the eigenvectors of the model (11)–(14) at each critical point will be necessary to find the boundary conditions to pass to the the traveling-wave calculator.

3.1.1 Critical point $(1, 0, 0, 0)$

The Jacobian Matrix of the functions on the RHS of the ODE (11)–(14) is

$$\begin{vmatrix} 0 & 1 & 0 & 0 \\ \frac{2}{D}u + \frac{1}{D}w - \frac{1}{D} & \frac{c}{D} & \frac{1}{D}u & 0 \\ 0 & 0 & 0 & 1 \\ -\alpha w & 0 & \alpha\beta - \alpha u & c \end{vmatrix}.$$

The eigenvalues of the function on the RHS of the ODE (11)–(14) at $(1, 0, 0, 0)$ are

$$\begin{aligned} \lambda_1 &= (c - \sqrt{c^2 + 4D})/2D, \\ \lambda_2 &= (c + \sqrt{c^2 + 4D})/2D, \\ \lambda_3 &= (c - \sqrt{c^2 - 4\alpha(1 - \beta)})/2, \\ \lambda_4 &= (c + \sqrt{c^2 - 4\alpha(1 - \beta)})/2. \end{aligned}$$

By analyzing these eigenvalues, certain restraints on parameters are obtained. First of all, $D \leq 1$. This will ensure that the exiting solutions do not fall into the $-w$ region. Since we are looking at a biologically interesting model, we do not want to deal with the situation of having negative amount of predators.

With the same goals in mind, there is the restraint $c > c^* = \sqrt{4\alpha(1 - \beta)}$. If this were not the case, then there would be 2 complex conjugate eigenvalues. This would cause a spiral around $(1, 0, 0, 0)$ which would lead to $w < 0$ as $s \rightarrow -\infty$. Figure 3 shows this scenario with $c < c^*$. The solutions oscillate and that forces the amount of predators to be less than 0.

The eigenvalues at $(1, 0, 0, 0)$, are all greater than 0 except for λ_1 . Therefore there is a 3-dimensional unstable manifold at $(1, 0, 0, 0)$.

3.1.2 Critical point $(\beta, 0, 1 - \beta, 0)$

Trying to solve for the eigenvalues at $(\beta, 0, 1 - \beta, 0)$ leads to the polynomial

$$p(\lambda) = \lambda^2(\lambda - c)(\lambda - c/D) - (\beta/D)\lambda(\lambda - c) + \alpha\beta(1 - \beta)/D.$$

Let

$$p(\lambda, D) = Dp(\lambda) = \lambda^2(\lambda - c)(D\lambda - c) - (\beta)\lambda(\lambda - c) + \alpha\beta(1 - \beta),$$

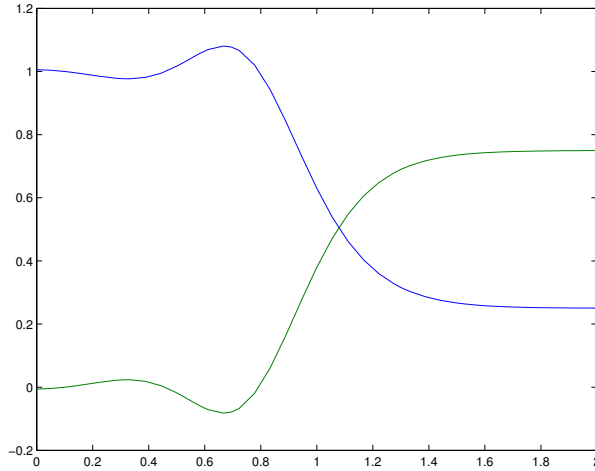


Figure 3: prey(blue) and pred(green) with $c < c^*$

$$p(\lambda, 1) = \lambda^2(\lambda - c)(\lambda - c) - (\beta)\lambda(\lambda - c) + \alpha\beta(1 - \beta).$$

Then there will be a local minimum when $\lambda_0 = (c \pm \sqrt{c^2 + 2\beta})/2$

$$p(\lambda_0, 1) = \alpha\beta(1 - \beta) - \beta^2/4,$$

therefore if $\alpha < \alpha^* = \frac{\beta}{4(1-\beta)}$ then $p(\lambda, 1)$ will have two negative real roots. Figure 4 shows the scenario of what will happen if $\alpha > \alpha^*$. The critical point $(\beta, 1 - \beta)$ is approached with damped oscillations because there are 2 imaginary stable eigenvalues. If $\alpha < \alpha^*$, then the critical point would be approached monotonely. For a more robust discussion on analyzing the value of α^* refer to Dunbar [D].

Also using the Hurwitz algorithm on $p(\lambda)$, we see that there will always be 2 negative real part eigenvalues and 2 positive real part eigenvalues. Therefore there is a 2-dimensional unstable manifold and a 2-dimensional stable manifold at $(\beta, 0, 1 - \beta, 0)$.

3.2 Calculating Traveling-Wave Front Solutions

3.2.1 Difficulties Finding Solutions

One problem when attempting to compute a solution that connects the two critical points is that these points are only reached when $s \rightarrow -\infty$ and $s \rightarrow +\infty$. Since computers do not have a value for ∞ , a solution must be found that connects some finite interval.

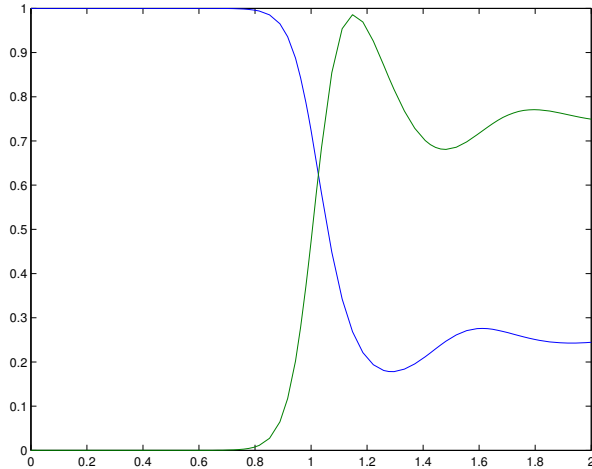


Figure 4: prey(blue) and pred(green) with $\alpha > \alpha^*$

Another problem is that the solution that connects the two points is very unstable. Let \mathbf{x} be the actual solution with \mathbf{x}_- , \mathbf{x}_+ as the stationary points $(1, 0, 0, 0)$ and $(\beta, 0, 1 - \beta, 0)$ respectively. Let n_{-u} be the number of dimensions for the unstable manifold at \mathbf{x}_- , n_{+s} be the number of dimensions for the stable manifold at \mathbf{x}_+ and n be the total number of dimensions in the system of equations. If $n_{-u} + n_{+s} = n + 1$ (which it is in our case, $3 + 2 = 4 + 1$), then there is only one solution. Combined with the limited amount of precision available on computers, using methods such as shooting and estimating points that should be on the solution are ineffective.

3.2.2 Boundary Value Problems

Boundary value problems are quite powerful for finding solutions to a first order system of ODE's and essential to the method used here to find the traveling-wave front solution. The most simple way to find solutions to a system of ODE's is by initial value problems (IVP). This is where one passes in the initial point of the system (also known as shooting). The problem with this is that it gives no information of the point where the solution should end up. In the case of finding a traveling-wave solution, this is an obvious drawback considering there is a point where the solution should end.

A boundary value problem (BVP) allows us to pass in information about the initial point of the solution and the end point of a solution. For example in an IVP, one would input $\mathbf{x}(\mathbf{0}) = [1 \ 0 \ 0 \ 0]$, then have Matlab compute a solution for a certain time span without giving any information about where

the solution should end. But with a BVP solver, one can pass a certain amount of conditions (that being the number of dimensions the system of equations is) about the starting point and the end point. It is possible to pass in $\mathbf{x}(\mathbf{0})_u = 1$, $\mathbf{x}(\mathbf{1})_w = \beta$ and also relations between points such as $\mathbf{x}(\mathbf{0})_v + \mathbf{x}(\mathbf{1})_z = 0$.

3.2.3 Splitting Up Solution

The method used to compute the traveling-wave solutions is to use a BVP solver with projection conditions and a phase condition. The interesting aspect of this method is that boundary conditions exist at 3 points of the solution. So in order to pass the boundary conditions into the BVP solver, the solution must be expressed as two parts (making the system of ODE's 8 dimensional).

Let \mathbf{y} be defined as the first section of \mathbf{x} such that $\mathbf{y}(\mathbf{0}) = \mathbf{x}(\mathbf{0})$ and \mathbf{z} be defined as the next section of \mathbf{x} such that $\mathbf{z}(\mathbf{1}) = \mathbf{x}(\mathbf{1})$. Then naturally $\mathbf{y}(\mathbf{1}) = \mathbf{z}(\mathbf{0})$. This will create 4 boundary conditions

$$\mathbf{y}(\mathbf{1}) - \mathbf{z}(\mathbf{0}) = \mathbf{0}.$$

3.2.4 Phase Condition

The next boundary condition is the phase condition. Let \mathbf{x}_d be an estimate of the derivative of \mathbf{x} at $\mathbf{y}(\mathbf{1})$, $\mathbf{z}(\mathbf{0})$ and \mathbf{x}_m be an estimate of $\mathbf{y}(\mathbf{1})$, $\mathbf{z}(\mathbf{0})$. Then the boundary condition is

$$\mathbf{x}_d^T \cdot (\mathbf{y}(\mathbf{1}) - \mathbf{x}_m) = 0.$$

While initially it may seem difficult to estimate a solution that we are looking for, this is not the case because of the eigenvalue and eigenvector analysis done in section 3.1. Since we know how manipulating the parameters will affect the shape of solution, we can choose parameters such that the solution will be simple enough to make estimations accurately. For example we know that if we choose an $\alpha < \alpha^*$, the solution will be a smooth curve, making it easier to estimate.

3.2.5 Projection Conditions

To find the last boundary conditions let \mathbf{A}_- and \mathbf{A}_+ be the matrix of eigenvectors (as columns) of the system of equations (11)–(14) computed at \mathbf{x}_- and \mathbf{x}_+ respectively. Let \mathbf{v}_{-s} be the stable eigenvector at \mathbf{x}_- and \mathbf{v}_{+u} be the

unstable eigenvectors at \mathbf{x}_+ . Then define \mathbf{L}_{-s} as a left stable eigenvector at \mathbf{x}_- and \mathbf{L}_{+u} as a left unstable eigenvector at \mathbf{x}_+ such that

$$\mathbf{L}_{-s} \cdot \mathbf{v}_{-s} = 1,$$

$$\mathbf{L}_{+u} \cdot \mathbf{v}_{+u} = 1.$$

The left eigenvalues can be found as row i of \mathbf{A}^{-1} where i is the corresponding eigenvector column in \mathbf{A} .

The final boundary conditions are

$$\mathbf{L}_{-s} \cdot (\mathbf{y}(0) - \mathbf{x}_-) = \mathbf{0},$$

$$\mathbf{L}_{+u} \cdot (\mathbf{z}(1) - \mathbf{x}_+) = \mathbf{0}.$$

3.3 Traveling-Wave Solution Code

Using the code I have provided to find traveling-wave solutions is quite simple. The only thing that needs to be changed is $b(\beta)$ in all three of the functions and the other parameters must be changed in `ppode` and `ppbc`. The `T` parameter in `ppode` (which is the magnitude of the time interval for the solution) will occasionally be too small and one will need to make it larger to see the whole traveling-wave solution under certain parameters. To better understand how the `bvp4c` function works and is implemented refer to Shampine, Kierzenka, Reichelt [SKR].

4 Conclusion

The goal of my research was to discover a way to compute traveling-wave solutions in the diffusive Lotka-Volterra pred-prey model and that is what I have attained. Many mathematical papers discuss traveling waves and the diffusive pred-prey model, yet they tend to be more theoretical. In my paper I have shown exactly how to view these solutions for this model by using Matlab (some papers discuss using AUTO to find traveling-wave solutions, but using Matlab is much more simple). This gives the opportunity to see exactly how changing the parameters affect the model. Although we have shown generally how affecting the parameters would affect the model, now one could get precise numbers on the effects.

4.1 Future Work

There are two pretty significant problems with the logistic growth model that was used in my paper. First of all there is a stable critical point that is approached as $t \rightarrow \infty$, which does not make much sense because it is unlikely that the predator and prey populations will stay at some certain amount

Also there should be some maximum amount of prey that the predators can eat. Both of these problems are resolved by adding another factor called the *satiation* effect. This is a more complicated model that could be examined.

Finally in the real world, the prey would tend to run away from the predators and the predators would in turn chase after the prey. This is another diffusion term that could be added to the model and analyzed.

4.2 Acknowledgements

To Gregory Lyng for his help and dedication throughout this project.

The code used to create the phase planes (figure 1 and 2) was written by John C. Polking, Rice University.

References

- [M] Murray, J.D., *Mathematical Biology*, Springer-Verlag, 1989.

- [D] Dunbar, S., *Traveling-Wave solutions of diffusive Lotka-Volterra equations: A heteroclinic connection in \mathbf{R}^4* , Transactions AMS, 286, no. 2, 1984, 557-594.

- [SKR] Shampine, L.F., Kierzenka J., Reichelt, M.W, *Solving Boundary Value Problems for Ordinary Differential Equations in Matlab with *bvp4c**, Lawrence F. Shampine, Jacek Kierzenka, Mark W. Reichelt. 2000

- [DF] Doedel, E.J., Friedman, M.J., *Numerical Computations of Heteroclinic Orbits* JCAM, 26, 1989, 155-170

- [B] Beyn, W.J., *The Numerical Computation of Connecting Orbits in Dynamical Systems*, IMA Journal of Numerical Analysis, 9, 1990, 379-405

Appendix

A Matlab Traveling-Wave Solutions Code

```
% This function finds traveling-wave solutions for the
% Pred-Prey model. To use it correctly, one must change
% the parameters in each function defined in this file.

function ppTravelingWaves

% enter parameters for pred-prey model
b = .25;
u0 = 1;
w0 = 0;
u1 = b;
w1 = 1-b;

% find traveling-wave solutions
solinit = bvpinit(linspace(0, 1, 10),
    [(u0+u1)/2 0 (w0+w1)/2 0 (u0+u1)/2 0 (w0+w1)/2 0]);
sol = bvp4c(@ppode,@ppbc,solinit);

% set up variables for graphing
eta = sol.x;
f = sol.y;

% append solutions
t = [eta (ones(1,size(eta))+eta)];
u = [f(1,:) f(5,:)];
v = [f(2,:) f(6,:)];
w = [f(3,:) f(7,:)];
z = [f(4,:) f(8,:)];

pp = [u; w]; %only plot pred and prey
%pp = [u; v; w; z]; %plot v and z in addition to pred and prey
```

```

% plot solutions
plot(t, pp);

%-----

% This function is the ode to be passed into the BVP solver
% y is [u v w z] and z is [u2 v2 w2 z2] as defined in the paper
function deriv = ppode(t, f)

% enter parameters for pred-prey model
a = .5;
b = .25;
c = 2;
d = 1;

% this is how far the solution will be calculated for
T = 100;

u = f(1);
v = f(2);
w = f(3);
z = f(4);
u2 = f(5);
v2 = f(6);
w2 = f(7);
z2 = f(8);

% split solution into two segments
u_prime = T*(v);
v_prime = T*((c/d)*v-(1/d)*u*(1-u-w));
w_prime = T*(z);
z_prime = T*(c*z-a*w*(u-b));
u2_prime = T*(v2);
v2_prime = T*((c/d)*v2-(1/d)*u2*(1-u2-w2));
w2_prime = T*(z2);
z2_prime = T*(c*z2-a*w2*(u2-b));

deriv = [u_prime;
         v_prime;
         w_prime;

```

```

        z_prime;
        u2_prime;
        v2_prime;
        w2_prime;
        z2_prime];

%-----

% This function calculates then returns the boundary conditions
function res = ppbc(ya, yb)

% enter parameters for pred-prey model
a = .5;
b = .25;
c = 2;
d = 1;

u0 = 1;
w0 = 0;
u1 = b;
w1 = 1-b;

% finding left eigenvectors for x(-infty)
% set up Jacobian Matrices
A0 = [0 1 0 0;
      ((2/d)*u0+w0/d-1/d) c/d u0/d 0;
      0 0 0 1;
      -a*w0 0 (a*b-a*u0) c];
A1 = [0 1 0 0;
      ((2/d)*u1+w1/d-1/d) c/d u1/d 0;
      0 0 0 1;
      -a*w1 0 (a*b-a*u1) c];

% compute eigenvectors
[E0, D0] = eig(A0);
[E1, D1] = eig(A1);

% get left eigenvectors
E0inv = E0^-1;
E1inv = E1^-1;

```

```

if D0(1,1) < 0;
    L0s = E0inv(1,:);
else
    disp('error: first eigenvalue is unstable');
end

if real(D1(1,1)) > 0 & real(D1(2,2)) > 0;
    L1u1 = E1inv(1,:);
    L1u2 = E1inv(2,:);
elseif real(D1(3,3)) > 0 & real(D1(4,4)) > 0;
    L1u1 = E1inv(3,:);
    L1u2 = E1inv(4,:);
elseif real(D1(2,2)) > 0 & real(D1(3,3)) > 0;
    L1u1 = E1inv(2,:);
    L1u2 = E1inv(3,:);
elseif real(D1(1,1)) > 0 & real(D1(4,4)) > 0;
    L1u1 = E1inv(1,:);
    L1u2 = E1inv(4,:);
else
    disp('error: corresponding roots not same sign');
end

% calculate estimate midpoint (xm)
% and estimate derivatives (xd)
xm = [(u0+u1)/2 0 (w0+w1)/2 0];
xd = [(u1-u0)/(2);0;(w1-w0)/(2);0];
xcom = xm*xd;
% calculate phase condition
phase = (xd(1)*yb(1)+xd(3)*yb(3)-xcom);

% calculate projection conditions
p0 =
L1u1(1)*yb(5)+L1u1(2)*yb(6)+L1u1(3)*yb(7)+L1u1(4)*yb(8)+L1u1*[-b;0;b-1;0];
p1 =
L1u2(1)*yb(5)+L1u2(2)*yb(6)+L1u2(3)*yb(7)+L1u2(4)*yb(8)+L1u2*[-b;0;b-1;0];
p2 = L0s(1)*ya(1)+L0s(2)*ya(2)+L0s(3)*ya(3)+L0s(4)*ya(4)+L0s*[-1;0;0;0];

% return the boundary conditions
res = [(yb(1) - ya(5))
       (yb(2) - ya(6))
       (yb(3) - ya(7))

```

```
(yb(4) - ya(8))  
p0  
p1  
p2  
phase];
```