Project 1 - A Mathematical Investigation of Populations and Predator-Prey Dynamics

Owen Hushka - Zach Alles - Zak Chehadi

July 29, 2023

1 Introduction

This paper examines various differential equations modeling predator-prey dynamics, specifically looking at populations of mountain lions and deer. First, we examine the Logistic Equation for single populations. Next, we use a Lotka-Volterra system to account for predator-prey interactions, using a system of differential equations with appropriate terms for positive growth (mountain lions) and negative growth (deer). This system assumed in the absence of interactions, mountain lions would exponentially decay, and deer would exponentially grow. Lastly, we looked at a slightly more sophisticated version of the Lotka-Volterra system, modifying the prey equation to account for environmental limitations on prey growth.

2 Modeling Individual Populations: The Logistic Equation

2.1 Equations examined

The first equation evaluated was the basic logistic equation. This was used to model the population growth of mountain lions, assuming that they are protected from hunting and have no natural predators. The equation modeling the growth of model lions is as follows:

$$\frac{dx}{dt} = r(1 - \frac{x}{L})x \tag{1}$$

This equation provides a reasonable model for population growth with $x(t)$ representing the size of the population (in dozens of animals) at any given time t (in years). The parameter $r > 0$ is the intrinsic growth rate and $L > 0$ is the carrying capacity of the population.

A logistic-type model for the deer population follows much of the same logic as the mountain lion equation, adding in a predation factor $H(x)$. Thus the logistic equation for the deer population is:

$$\frac{dx}{dt} = r(1 - \frac{x}{L})x - H(x) \tag{2}$$

with $H(x)$ being:

$$H(x) = \frac{px^2}{q + x^2} \tag{3}$$

The parameters p and q represent the skill of the mountain lions in hunting while the other variables are the same as in equation (1).

2.2 Using the Logistic Models: Question Set A

The units of r and L for this model are dozens of animals/year and dozens of animals, respectively. The equilibrium solutions to equation (1) are found by setting the right side $r(1 - \frac{x}{L})x = 0$ which yields the answers $x = 0$ and $x = L$. To solve for $x(t)$ explicitly, separation of variables and partial fraction decomposition yields (steps shown in Appendix A):

$$x(t) = \frac{L}{1 - (1 - \frac{L}{x_0})e^{-rt}} \tag{4}$$

Using the values of $r_m = 0.65$ and $L_m = 5.4$ over the interval of $t \in [0, 30]$ with step sizes of $h = 0.5$, $h = 0.1$, and $h = .01$ with Euler's Method produces the following plot of Mountain Lion Growth curves.

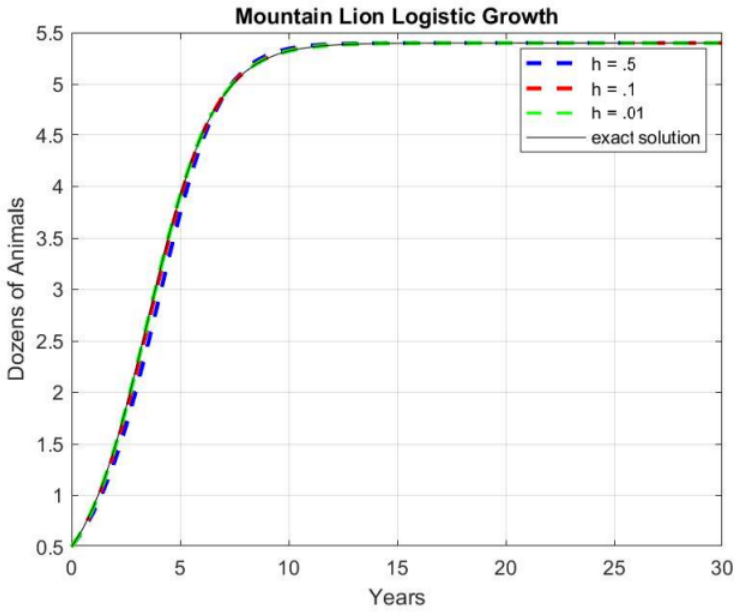

Figure 1: Euler's Method with Eq. (1) with various step sizes

All three models offer a decent approximation of the logistic equation. The step size with $h = .01$ (green) most closely follows the exact solution (black). The step size with $h = .5$ is the least accurate. It is easy to see that as the step size decreases, accuracy increases.

Also graphed below is the semi-log graph of the absolute error given by $AbsoluteError = |ExactSolution - ApproximateSolution|$ (Figure 2). In general, the absolute error decreases as all models approach the carrying capacity of the population model. This would make sense as all models should approach $x(t) = 5.4(dozen)$ because that is what the model states the carrying capacity is, which can be understood as the total number of animals that can be supported by the system.

It's also interesting to note all three approximations have downward spikes in error around $t = 7$. This can be explained as crossing a threshold for when the models change from underestimates to overestimates of the actual solution. To further illustrate this point, see figure 10 in Appendix C, which zooms in on the Mountain Lion Logistic equation near $t = 7$.

Based on the error curves, the best step size to balance accuracy with efficiency would be $h = .1$. This step size has less error than $h = .5$ but only requires 301 calculations versus the 3001 calculations required by step size $h = .01$. The gains in accuracy from the extra calculations with step size $h = .01$ don't increase the accuracy so much as to justify the increased workload.

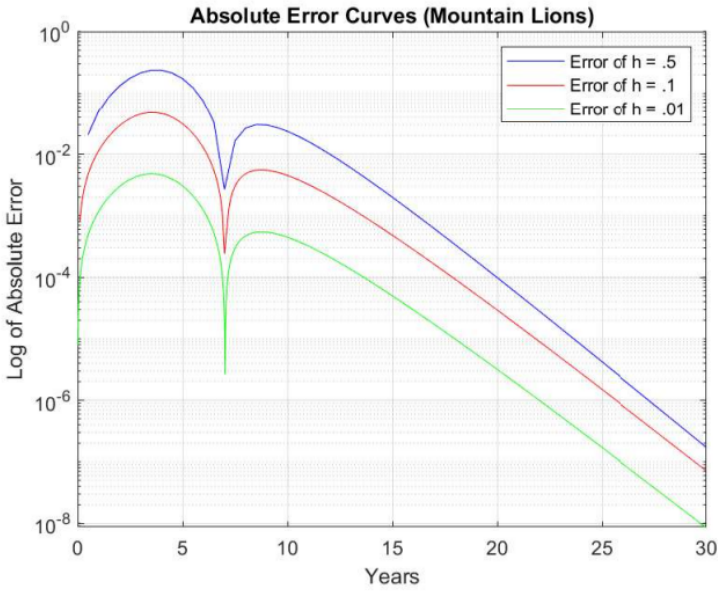

Figure 2: Absolute Error of the Euler Approximations

In examining equation (2), it is noted that the equation is 1^{st} order, non-linear and autonomous. The highest derivative of $x(t)$ is the first derivative making it a first-order equation. The x^2 terms make the equation non-linear. Finally, there is no explicit expression of t in the equation making it autonomous.

Looking at the harvesting term, $H(x) = \frac{px^2}{q+x^2}$, if you take the limit as $x \to \infty$ we can see that $H(x) \to p$. This means the maximum the harvest term could be is p. This can be understood physically as no matter how large the deer population becomes, the mountain lions will only become so skilled at hunting them. Taking the limit as the population $x \to 0$, the term $H(x) \to 0$. This also makes sense because if the deer population dwindles toward extinction, there won't be any deer left to hunt.

The positive equilibrium solutions to equation (2) up to four decimal places are $x = 0.0000$, $x = 0.8018$, $x = 1.8564$, and $x = 5.4418$. The code to find these can be found in Appendix B.

Plotted below for equation (2) are the equilibrium solutions along with four Euler's approximations with different starting population sizes of 6, 12, 18, and 84 animals. These solutions are plotted on top of the direction field for equation (2). For this equation, the growth rate $r_d = 0.65$, the carrying capacity $L_d = 8.1$, $p = 1.2$ and $q = 1$.

The plot shows that population growth trajectories depends on the total initial population. $x = 0.8018$ dozen animals is a stable solution and any initial population on the interval $x \in (0, 1.8565)$ will tend toward this solution. $x = 1.8564$ is an unstable solution, so any starting population near this value will tend to increase or decrease, if the population is greater than or less than 1.8564 dozen animals, respectively. The model shows that $x = 5.4418$ is a stable solution and the solutions curves on the interval $x \in (1.8564, \infty)$ will tend toward 5.4418 dozen animals.

These phenomena may be explained as the ability of deer populations to recover from predation from the mountain lions given an initial population. If the population starts below 1.8564 dozen animals it can never grow beyond 1.8564 because it cannot maintain a large enough population for continued growth. It will always tend towards 0.8018 dozen animals as the mountain lions hunting deer and the deer reproducing will be in equilibrium. If it starts higher than 1.8564 but less than 5.4418 dozen animals it will grow towards 5.4418 dozen animals. But if it starts higher than 5.4418 dozen animals, the population will always decrease as the animals are hunted faster than they can reproduce. It will never be able to reach the carrying capacity of 8.1 so long as mountain lions are present.

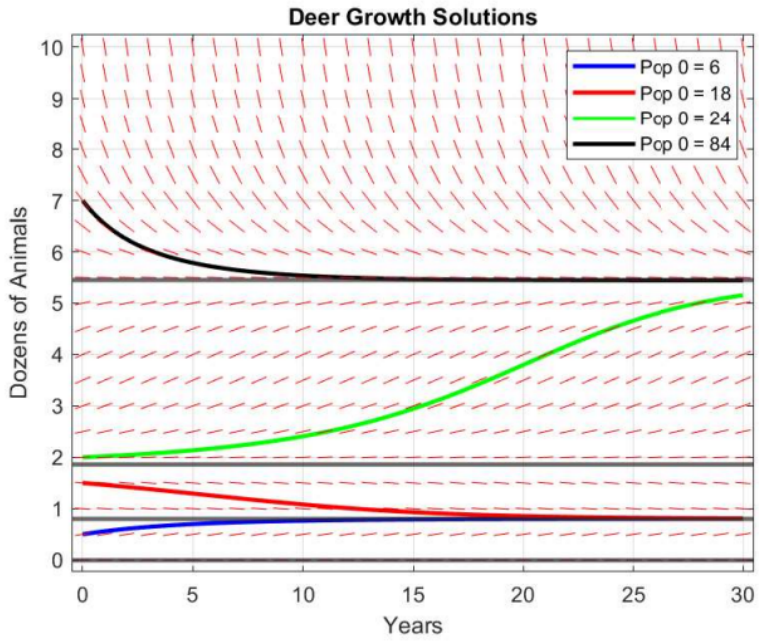

Figure 3: Euler Deer growth solutions with different starting populations

3 Modeling Population Interactions

3.1 Lotka-Volterra System

One of the simplest models of predator-prey interactions is the Lotka – Volterra system

$$\frac{dx_1}{dt} = -\alpha x_1 + \beta x_1 x_2 \tag{5}$$

$$\frac{dx_2}{dt} = \gamma x_2 - \delta x_1 x_2 \tag{6}$$

3.2 Question Set B

3.2.1 Classify the Lotka-Volterra system. Is it linear? Autonomous? What is its order?

The system is non-linear. This is because we have the $x_1 x_2$ term. It is autonomous since there is no t term in either function. It is first order.

3.2.2 Analytically find the v and h nullclines and all equilibrium solutions of. DO NOT use specific values for any of the parameters.

$-\alpha x_1 + \beta x_1 x_2 = 0$
$x_1(\beta x_2 - \alpha) = 0$

$$x_1 = 0 \tag{7}$$

$$x_2 = \frac{\alpha}{\beta} \tag{8}$$

$\gamma x_2 - \delta x_1 x_2 = 0$
$x_2(\gamma - \delta x_1) = 0$

$$x_2 = 0 \tag{9}$$

$$x_1 = \frac{\gamma}{\delta} \tag{10}$$

Equilibrium solutions: $(0, 0)$ and $(\frac{\gamma}{\delta}, \frac{\alpha}{\beta})$

3.2.3 Plot the vector field and the nullclines on the x_1x_2-plane

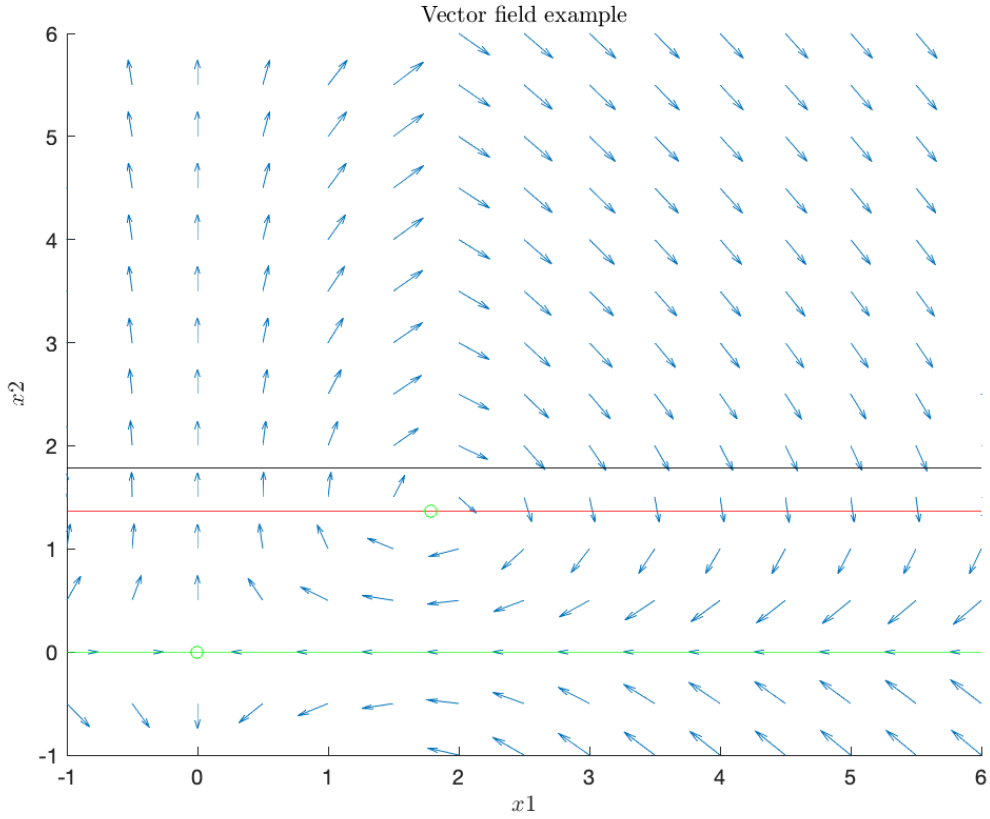

Figure 4: Vector Field With Nullclines and Equilibrium Solutions

3.2.4 Use ode45 to simulate solutions starting from the initial condition $x_1 = 0.5$ and $x_2 = 1$. Plot the vector field together with the trajectory. Does the solution curve behave as expected? Why or why not?

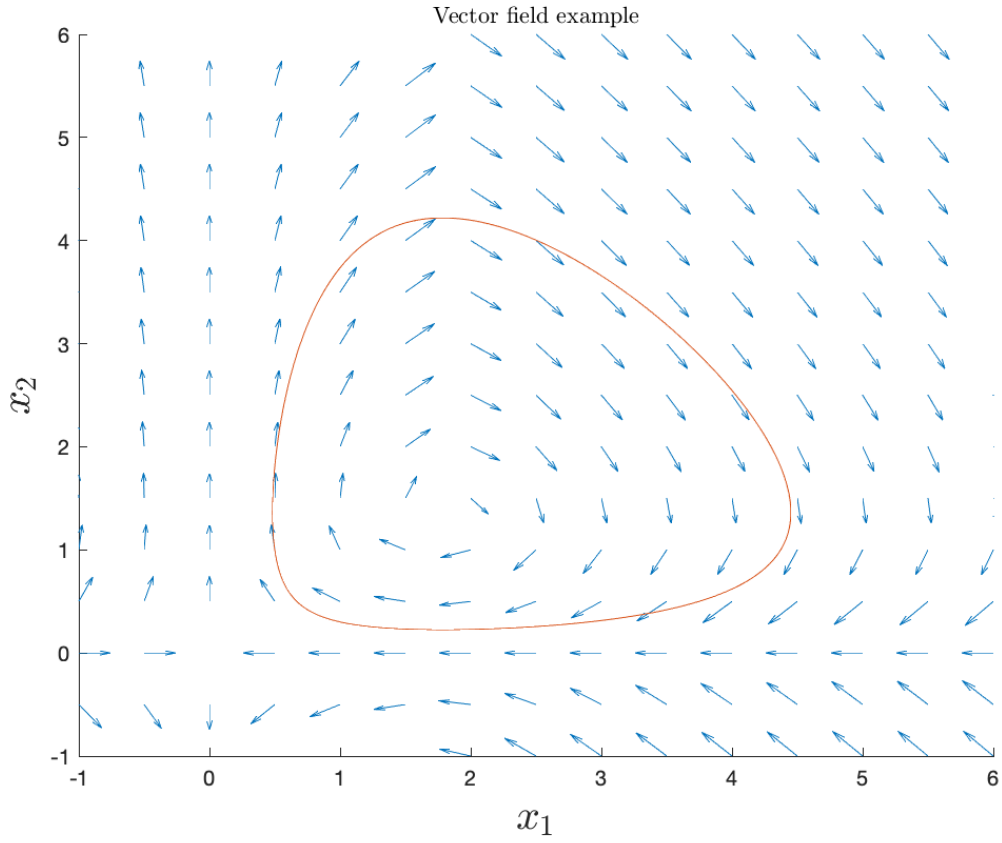

Figure 5: Vector Field with specific solutions

This solution curve behaves as expected. The vector field shows us how we would expect it to behave and the curve follows the vector field. This curve also follows what one would intuitively expect the graph to look like. A cyclical curve where the prey population grows, the predator population has more food and begins to grow as well, the prey population is being hunted more and begins to decline, the predator population has less food so it begins to decline and then the cycle restarts with the prey having fewer predators and thus the prey's population increases.

3.2.5 Use ode45 in Matlab to simulate solutions to the system of equations starting from the same initial conditions. Plot the component curves $x_1(t)$ and $x_2(t)$ together against t. Discuss these plots. Are the curves in phase or out of phase? What does this mean physically in terms of predator-prey interactions? Note that the "component curve" solutions are given by all pairs of points (t,x_1(t)) and (t,x_2(t)).

These curves are in phase. Physically, in terms of predator-prey interactions, it makes sense that we see these two populations in phase together. We first see the prey population rise. This is in turn followed by the predator population growing since there is more food available to them. As there begins to be more predators eating the prey, the prey population begins to drop. As the prey population drops, there is less food for the predators and the predator population begins to fall. With a lower predator population, we would expect the prey's population to begin to grow again and this is exactly what happens. The cycle starts over again.

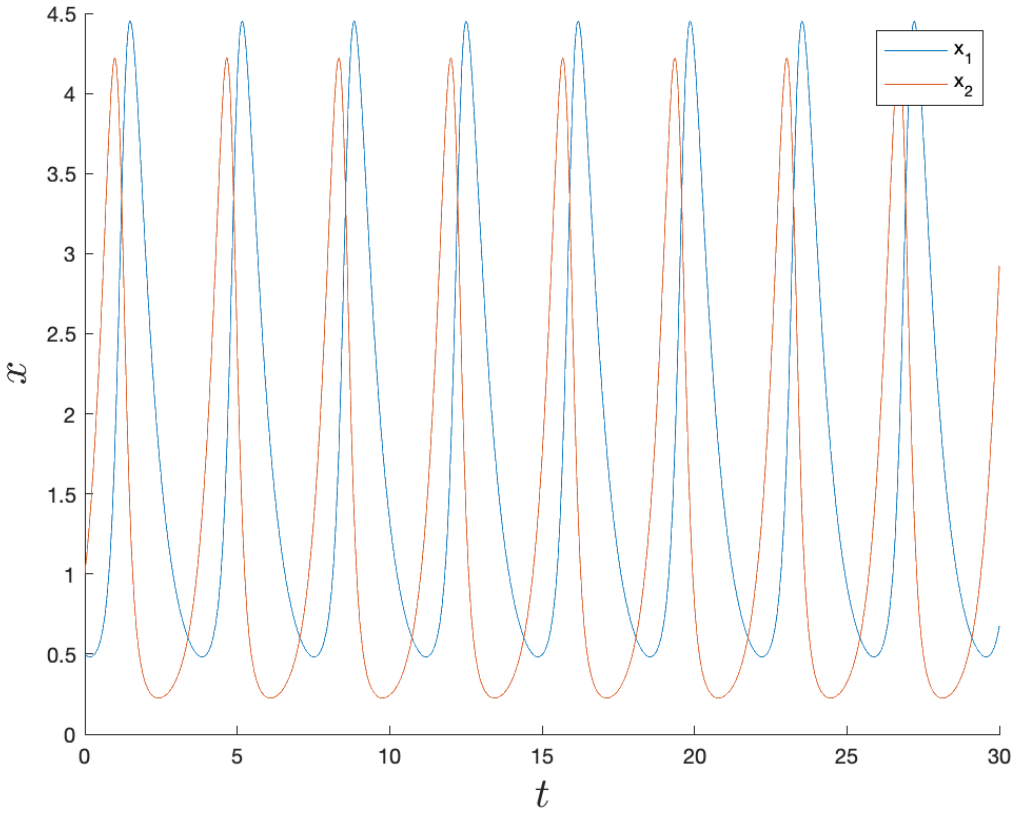

Figure 6: Predator and Prey Populations Graphed Together

3.3 The Logistic Predator-Prey Equations

Recall that an underlying assumption of (5)(6) is that both species will exhibit exponential behavior if there are no inter-species interactions, that is, if the interaction terms x_1x_2 are excluded from the model. This assumption ignores the natural limits imposed on a prey population by its environment, such as finite food. A slightly more complicated but realistic way to model predator-prey interactions is to adjust the prey equation x_2 to include these constraints, which gives rise to the Logistic Predator-Prey system

$$\frac{dx_1}{dt} = -\alpha x_1 + \beta x_1 x_2 \tag{11}$$

$$\frac{dx_2}{dt} = \gamma(1 - \kappa x_2)x_2 - \delta x_1 x_2 \tag{12}$$

Now the underlying assumption of this model is, in the absence of predation, the prey population will obey a logistic growth model instead of an exponential growth model. This means that

- If the prey population is small, the rate of growth is approximately proportional to its size
- If the prey population is too large to be supported by its environment, the rate of change of the population will decrease

As in (5)(6), if the species do not interact the predator population will exhibit exponentially decaying solutions.

3.4 Question Set C

3.4.1 Analytically find the v and h nullclines and equilibrium solutions of the Logistic Predator-Prey system (11)(12).

Need to set $\frac{dx_1}{dt} = 0 and \frac{dx_2}{dt} = 0$

To find the V-nullclines:

$0 = -\alpha x_1 + \beta x_1 x_2$

$\alpha x_1 = \beta x_1 x_2$

$0 = x_1$

$\frac{\alpha}{\beta} = x_2$

To find the H-nullclines:

$0 = \gamma(1 - \kappa x_2)x_2 - \delta x_1 x_2$

$\gamma(1 - \kappa x_2)x_2 = \delta x_1 x_2$

$x_2 = 0$

$\delta x_1 = \gamma(1 - \kappa x_2)$

$x_1 = \frac{\gamma(1-\kappa x_2)}{\delta}$

Equilibrium Solutions:

$(0, 0)$

$(0, \frac{1}{\kappa})$

$(\frac{\gamma(1-\kappa(\frac{\alpha}{\beta}))}{\delta}, \frac{\alpha}{\beta})$

3.4.2 Assign the parameters in (11)(12) the values $\alpha = 1.5, \beta = 1.1, \gamma = 2.5, \delta = 1.4, and \kappa = 0.5$. Modify the script flow.m in Matlab so it computes the vector field of the system (11)(12). Use ode45 in Matlab to simulate solutions to (11)(12) starting from the initial conditions (x1(0),x2(0)) = (5,1) and (x1(0),x2(0)) = (1,5) over the time interval t ϵ [0,30] and then, using the color scheme of Question Set B number 3, plot the following together:

(a) Use flow.m to plot the vector field of (11)(12) on the domain $-1 < x_1 < 6, 1 < x_2 < 6$
(b) Plot the nullclines and the equilibrium solutions of (11)(12) on the domain $1 < x_1 < 6, 1 < x_2 < 6$
(c) Plot the trajectories (x1,x2) on the domain $1 < x_1 < 6, 1 < x_2 < 6$

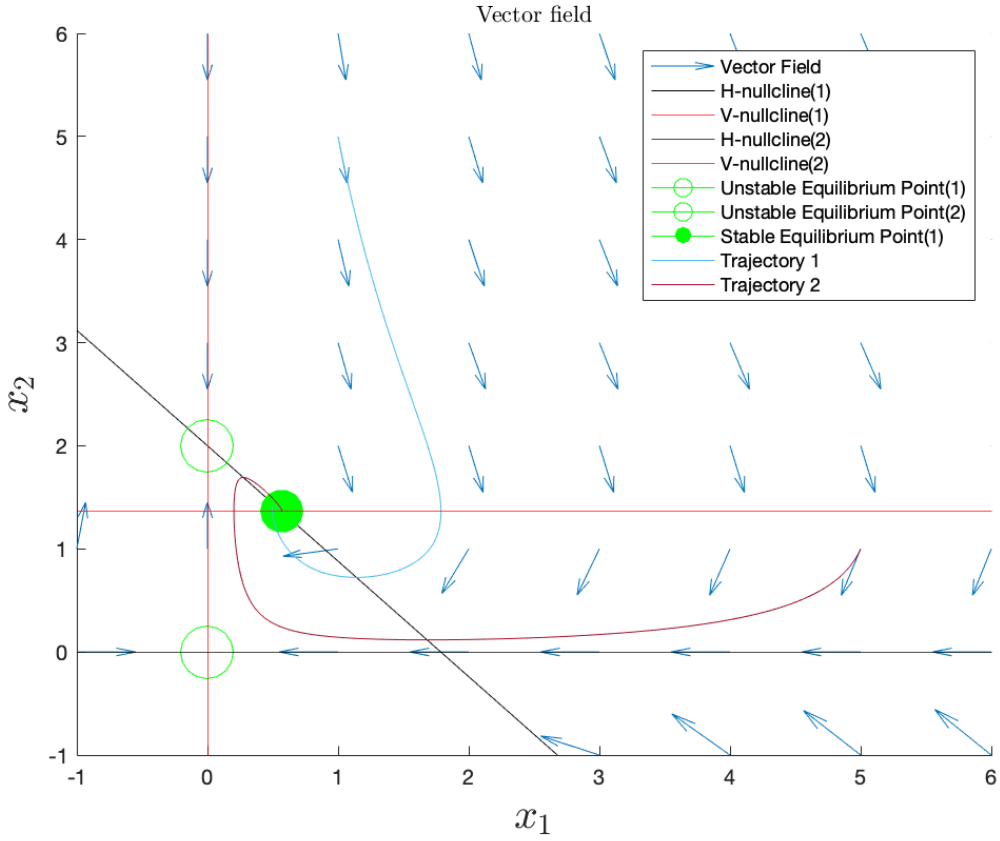

Figure 7: Vector Field With Nullclines and Equilibrium Solutions

This Plot shows that the equilibrium point at approx. (0.75,1.4) is in fact stable, because both trajectory curves point towards it.

3.4.3 Assign the parameters in (5) the values $\alpha = 1.5, \beta = 1.1, \gamma = 2.5, \delta = 1.4, and \kappa = 0.5$.. Use ode45 in Matlab to simulate solutions to (5) starting from the initial conditions (x1(0),x2(0)) = (5,1) and (x1(0),x2(0)) = (1,5) over the time interval t ϵ [0,30]. Plot the component curves $x_1 and x_2$ together against t, including a legend. Make a separate plot for each of the initial conditions. Discuss the solution curves. Are they periodic? Is there asymptotic behavior?

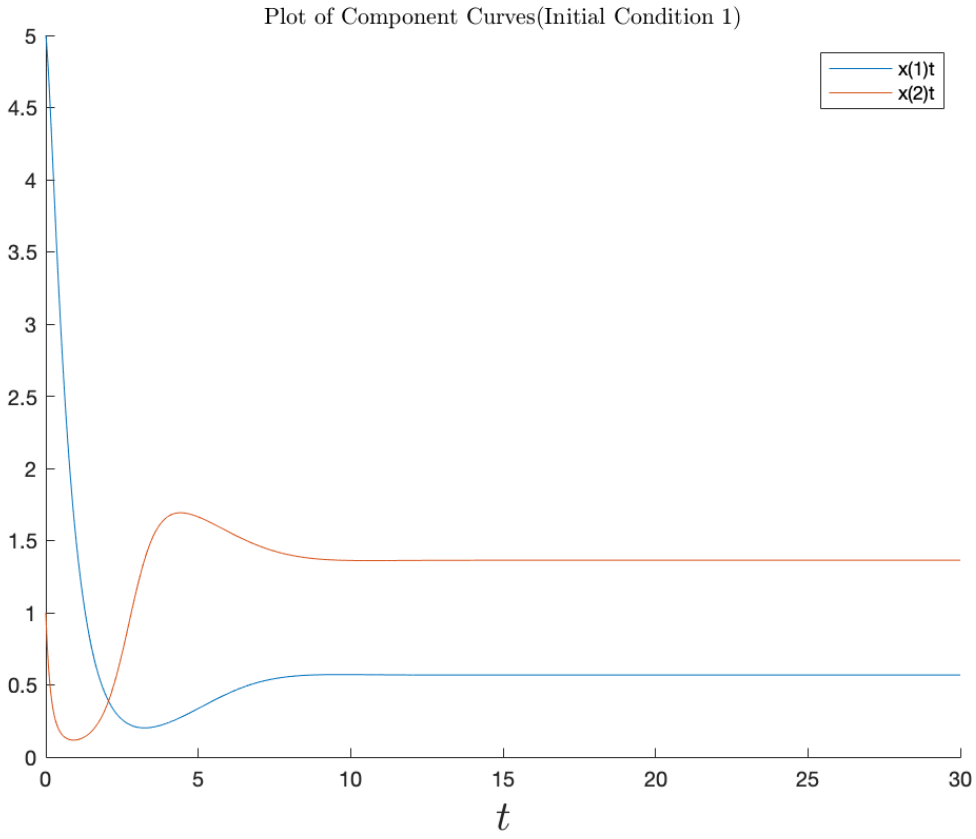

Figure 8: Component Curve 1

Both solution curves are not periodic as they don't continuously repeat. However, both curves have asymptotic behavior. The first component curve has vertical asymptotes at t = 0 that go to infinity, and there are also two horizontal asymptotes. x_1 stabilizes at around 0.5 and x_2 stabilizes at about 1.4. The second component curve has one vertical asymptote at t = 0 where x_2 goes to infinity. It also has two horizontal asymptotes. x_1 evens out at around 0.5 and x_2 at around 1.35.

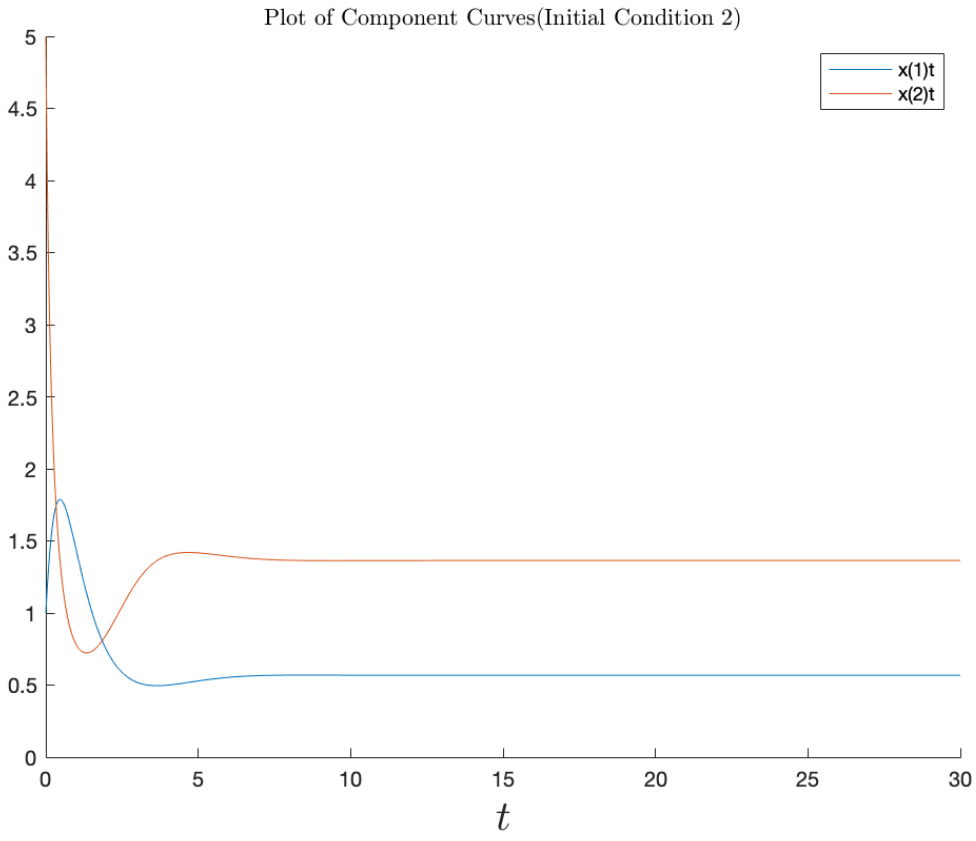

Figure 9: Component Curve 2

4 Model Comparison

One difference between the models is that the Lotka-Volterra System is periodic and the Predator-Prey equations are asymptotic. So over an infinite amount of time in the Predator-Prey system, the populations of the animals will plateau, but in the Lotka-Volterra System, the populations will cycle up and down. The strengths of Lotka-Volterra System are that it is more accurate in how the populations will keep varying over time, and the strength of the Predator-Prey system is that it includes other factors such as food and the animals' surroundings. The inclusion of these other factors makes the Predator-Prey System more accurate in terms of predicting the population at the start.

The main Weakness of the Lotka-Volterra system is that it ignores the natural limits of the prey population, such as food resources. A weakness present in both models is the fact that the predator equation is simply reliant on the mortality rate of the predator species and the conversion rate of food into offspring. To enhance both of the models, it would be helpful to take into account additional factors for the predators, such as population decline due to habitat loss, hunting, or competition with other predators. All of these factors are ignored in the models and could play an important role in the decline of the population. Another factor that would make the predator-prey equations more accurate would be to add a term that would account for the availability of different food sources, such as other prey or human sources of food i.e. human trash and/or leftovers near predator habitats. Predator populations may grow beyond what the model predicts if there are outside factors contributing to the caloric intake of the predator. The same could be said for the prey populations. Perhaps a larger system of differential equations accounting for multiple predators and prey species would be the most accurate method to model population growth, albeit at the expense of complicating the analysis.

5 Appendix A (Mathematical steps shown):

Deriving Explicit x(t) for Equation (1):

$$\frac{dx}{dt} = r(1 - \frac{x}{L})x$$

Separation of Variables yields

$$\int \frac{dx}{x(1 - \frac{x}{L})} = -\int r dt = -rt + c$$

$$\int \frac{1}{x - L}\, dx \; - \int \frac{1}{x}\, dx = -rt + c$$

$$\ln(x - L) - \ln x = -rt + c$$

$$e^{ln|\frac{x-L}{x}|} = e^{-rt+c}$$

$$1 - \frac{L}{x} = ce^{-rt}$$

$$x(t) = \frac{L}{1 - ce^{-rt}}$$

Apply initial condition $x(0) = x_0$

$$x_0 = \frac{L}{1 - ce^0}$$

$$1 - c = \frac{L}{x_0}$$

$$c = 1 - \frac{L}{x_0}$$

$$x(t) = \frac{L}{1 - (1 - \frac{L}{x_0})e^{-rt}}$$

Work shown for the limit of $H(x) as x \to \infty$

$$\lim_{x\to\infty} H(x) = \lim_{x\to\infty} \frac{px^2}{q + x^2}$$

Apply L'Hopital's Rule

$$\lim_{x\to\infty} \frac{2px}{2x} = p$$

Work shown for the limit of H(x) as $x \to 0$

$$\lim_{x\to 0} H(x) = \lim_{x\to 0} = \frac{px^2}{q + x^2} = \frac{0}{q} = 0$$

6 Appendix B: (Code)

6.1 Code for Figures 1 and 2

```
% Euler's Method
% Initial conditions and setup
h1 = .5; % step size
h2 = .1;
h3 = .01;

t1 = 0:h1:30;  % the range of x
x1 = zeros(size(t1));  % allocate the result x
x1(1) = .5; % the initial x value
n = numel(x1);  % the number of x values

t2 = 0:h2:30;
x2 = zeros(size(t2));
x2(1) = .5;
n2 = numel(x2);

t3 = 0:h3:30;
x3 = zeros(size(t3));
x3(1) = .5;
n3 = numel(x3);


% The loop to solve the DE
for i = 1:n-1
    f = .65*x1(i) - (.65/5.4)*x1(i).^2;
    x1(i+1) = x1(i) + h1 * f;
end

for i2 = 1:n2-1    %Loop for H2
    f = .65*x2(i2) - (.65/5.4)*x2(i2).^2;
    x2(i2+1) = x2(i2) + h2 * f;
end


for i3 = 1:n3-1       % LOOP for H3
    f = .65*x3(i3) - (.65/5.4)*x3(i3).^2;
    x3(i3+1) = x3(i3) + h3 * f;
end

figure(1)
plot(t1,x1,"b--","LineWidth",2);
hold on
plot (t2,x2,"r--","LineWidth",2);
plot (t3,x3,"g--","LineWidth",2);

v = linspace(0,30,61);
u = 5.4*(1-(1-5.4/.5)*exp(-.65*v)).^-1;
plot(v,u,"k-")
legend("h = .5","h = .1","h = .01","exact solution")
```

```
xlabel("Years")
ylabel("Dozens of Animals")
title("Mountain Lion Logistic Growth")
grid on;
hold off;

%ABSOLUTE ERROR PLOT Y values will abserror, x value will be t4

v2 = linspace(0,30,301);
u2 = 5.4*(1-(1-5.4/.5)*exp(-.65*v2)).^-1;

v3 = linspace(0,30,3001);
u3 = 5.4*(1-(1-5.4/.5)*exp(-.65*v3)).^-1;

abserror1 = abs(u - x1);
abserror2 = abs(u2 - x2);
abserror3 = abs(u3 - x3);


figure(2)
semilogy(t1,abserror1,"b-");
hold on
semilogy(t2,abserror2,"r-");
hold on
semilogy(t3,abserror3,"g-");
legend("Error of h = .5","Error of h = .1","Error of h = .01")
xlabel("Years")
ylabel("Log of Absolute Error")
title("Absolute Error Curves (Mountain Lions)")
grid on
hold off
```

6.2 Code for Equation 2 zeros

```
clear all;
close all;

x = linspace(0,8,600);
dxdt = .65*x.*(1 - (x./8.1)) - ((1.2*(x.^2))./(1 + (x.^2)));

plot(x,dxdt,"r-");
grid on;


fun = @fq1;
x0 = .01;
x1 = [.5 1];
x2 = [1.5 2];
x3 = [5 6];
z1 = fzero(fun,x0);
z2 = fzero(fun,x1);
z3 = fzero(fun,x2);
z4 = fzero(fun,x3);
```

```
function y = fq1(x)
y = .65.*x.*(1 - (x./8.1)) - ((1.2*(x.^2))./(1 + (x.^2)));
end
```

6.3 Code for Figure 3

```
function dirfield(f,tval,yval,plot_title)

% dirfield(f, t1:dt:t2, y1:dy:y2)
%
%   plot direction field for first order ODE y' = f(t,y)
%   using t-values from t1 to t2 with spacing of dt
%   using y-values from y1 to t2 with spacing of dy
%
%   f is an @ function, or an inline function,
%     or the name of an m-file with quotes.
%
% Example: y' = -y^2 + t
%   Show direction field for t in [-1,3], y in [-2,2], use
%   spacing of .2 for both t and y:
%
%   f = @(t,y) -y^2+t


[tm,ym]=meshgrid(tval,yval);
dt = tval(2) - tval(1);
dy = yval(2) - yval(1);
fv = vectorize(f);
if isa(f,'function_handle')
  fv = eval(fv);
end
yp=feval(fv,tm,ym);
s = 1./max(1/dt,abs(yp)./dy)*0.35;
h = ishold;
quiver(tval,yval,s,s.*yp,0,'.r'); hold on;
quiver(tval,yval,-s,-s.*yp,0,'.r');
if h
  hold on
else
  hold off
end
axis([tval(1)-dt/2,tval(end)+dt/2,yval(1)-dy/2,yval(end)+dy/2])

title(plot_title);
xlabel('t values');
ylabel('y values');
end
```

```
close all; clear all;

% Euler's Method
% Initial conditions and setup
h = .1; % step size
t = 0:h:30;  % the range of x 6 ANIMALS

x1 = zeros(size(t));  % allocate the result x
x1(1) = .5; % the initial x value 6 ANIMALS
n = numel(x1);  % the number of x values


x2 = zeros(size(t));
x2(1) = 1.5; % 18 ANIMALS
n2 = numel(x2);

x3 = zeros(size(t));
x3(1) = 2; % 24 ANIMALS
n3 = numel(x3);

x4 = zeros(size(t));
x4(1) = 7; % 84 ANIMALS
n4 = numel(x4);


% The loop to solve the DE FOR 6 ANIMALS
for i = 1:n-1
    f = .65*x1(i).*(1 - (x1(i)./8.1)) - ((1.2*(x1(i).^2))./(1 + (x1(i).^2)));
    x1(i+1) = x1(i) + h * f;
end

for i2 = 1:n2-1    %Loop for 18 ANIMALS
    f = .65*x2(i2).*(1 - (x2(i2)./8.1)) - ((1.2*(x2(i2).^2))./(1 + (x2(i2).^2)));
    x2(i2+1) = x2(i2) + h * f;
end


for i3 = 1:n3-1       % LOOP for 24 ANIMALS
    f = .65*x3(i3).*(1 - (x3(i3)./8.1)) - ((1.2*(x3(i3).^2))./(1 + (x3(i3).^2)));
    x3(i3+1) = x3(i3) + h * f;
end

for i4 = 1:n3-1       % LOOP for 24 ANIMALS
    f = .65*x4(i4).*(1 - (x4(i4)./8.1)) - ((1.2*(x4(i4).^2))./(1 + (x4(i4).^2)));
    x4(i4+1) = x4(i4) + h * f;
end

plot(t,x1,"b-",'linewidth', 2);
hold on;
```

```
plot (t,x2,"r-",'linewidth',2);
plot (t,x3,"g-",'linewidth',2);
plot (t,x4,"k",'linewidth',2);
yline(0.8018,"LineWidth",2)
yline(0,"LineWidth",2)
yline(1.8564,"LineWidth",2)
yline(5.4418,"LineWidth",2)

f = @(t,x) .65.*x.*(1 - (x./8.1)) - ((1.2*(x.^2))./(1 + (x.^2)));
dirfield(f,0:1:30,0:.5:10,"Direction Field for Deer Population");

legend("Pop 0 = 6","Pop 0 = 18","Pop 0 = 24","Pop 0 = 84 ")
xlabel("Years")
ylabel("Dozens of Animals")
title("Deer Growth Solutions")
grid on;
hold off;
```

6.4 Code for Figure 4

```
close all; clear all;
% This Matlab code generates a vector field for the system of ODEs
% dx1/dt = f(x1,x2), dx2/dt = g(x1,x2)
% This code currently will find the vector field for the EXAMPLE problem
%           dx1/dt = a*x2
%           dx2/dt = -x1
%-------------------------------------------------------------------------
%        THESE ARE NOT THE PROBLEMS YOU ARE SOLVING FOR PROJECT 1!
% (To have this code generate the vector fields for the Project 1 systems
% of equations, make any necessary adjustments in the sections of code
% labeled with "Step i" where i = 1, 2, 3, 4, or 5)
%-------------------------------------------------------------------------
% Step 1: Set the axis limits so that you plot the vector field over the
%         intervals x1min < x1 < x1max, x2min < x2 < x2max
    x1min = -1; x1max = 6; x2min = -1; x2max = 6;
% Step 2: pick step sizes for x1 and x2;
    x1step = 0.5; x2step = 0.5;
% generate mesh for plotting
    [x1, x2] = meshgrid(x1min:x1step:x1max, x2min:x2step:x2max);
% Step 3: define all needed parameter values
    a = 1.5;
    b = 1.1;
    y = 2.5;
    z = 1.4;
% Step 4: define the system of equations you are using
    dx1 = -a*x1 + b*x1.*x2;
    dx2 =  y*x2 - z*x1.*x2;
% normalize vectors (to help plotting)
    dx1 = dx1./sqrt(dx1.^2 + dx2.^2);
    dx2 = dx2./sqrt(dx1.^2 + dx2.^2);
% generate the vector field
    hold on
    quiver(x1, x2, dx1,dx2,'AutoScaleFactor',0.5);
    x2=a/b;
    fplot(x2,'r')
    x2 = 0;
    fplot(x2, 'k');
    x1=y/z;
    fplot(x1, 'k');
    x1 = 0;
    fplot(x1, 'g');
    plot(0,0,'og');
    plot(y/z, a/b, 'go');
    hold off
% specify the plotting axes
    axis([x1min x1max x2min x2max])
% Step 5: label the axes, include a title
    xlabel('$x1$','Interpreter','latex')
    ylabel('$x2$','Interpreter','latex')
    title('Vector field example','Interpreter','latex')
```

6.5 Code for Figure 5

```
close all; clear all;
% This Matlab code generates a vector field for the system of ODEs
% dx1/dt = f(x1,x2), dx2/dt = g(x1,x2)
% This code currently will find the vector field for the EXAMPLE problem
%           dx1/dt = a*x2
%           dx2/dt = -x1
%-------------------------------------------------------------------------
%        THESE ARE NOT THE PROBLEMS YOU ARE SOLVING FOR PROJECT 1!
% (To have this code generate the vector fields for the Project 1 systems
% of equations, make any necessary adjustments in the sections of code
% labeled with "Step i" where i = 1, 2, 3, 4, or 5)
%-------------------------------------------------------------------------
% Step 1: Set the axis limits so that you plot the vector field over the
%         intervals x1min < x1 < x1max, x2min < x2 < x2max
    x1min = -1; x1max = 6; x2min = -1; x2max = 6;
% Step 2: pick step sizes for x1 and x2;
    x1step = 0.5; x2step = 0.5;
% generate mesh for plotting
    [x1, x2] = meshgrid(x1min:x1step:x1max, x2min:x2step:x2max);
% Step 3: define all needed parameter values
    a = 1.5;
    b = 1.1;
    y = 2.5;
    z = 1.4;
% Step 4: define the system of equations you are using
    dx1 = -a*x1 + b*x1.*x2;
    dx2 =  y*x2 - z*x1.*x2;
% normalize vectors (to help plotting)
    dx1 = dx1./sqrt(dx1.^2 + dx2.^2);
    dx2 = dx2./sqrt(dx1.^2 + dx2.^2);
% generate the vector field
    hold on
    quiver(x1, x2, dx1,dx2,'AutoScaleFactor',0.5);

% specify the plotting axes
    axis([x1min x1max x2min x2max])
% Step 5: label the axes, include a title
    xlabel('$x1$','Interpreter','latex')
    ylabel('$x2$','Interpreter','latex')
    title('Vector field example','Interpreter','latex')



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rudimentary script to solve an initial value problem consisting of
% the two example differential equations here on the interval a<=t<=b
%
% x1'(t) = alpha*x1+beta*x2,  x1(a) = x1_0
% x2'(t) = gamma*x1+delta*x2, x2(a) = x2_0
%
% Look for "*** Your entry i ***" where i=1,2,3,4,5,6 to make appropriate
% changes to solve your specific system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Housekeeping


    % ODE45 options
```

```
options = odeset('AbsTol',1e-10,'RelTol',1e-7);

    % Independent variable

a = 0;      % left endpoint of t interval     *** Your entry 1 ***
b = 30;   % right endpoint of t interval    *** Your entry 2 ***
h = 0.01;   % stepsize                        *** Your entry 3 ***
tt = a:h:b;

    % Initial value of dependent variables

x1_0 = 0.5;    % initial value of x1    *** Your entry 4 ***
x2_0 = 1;    % initial value of x2    *** Your entry 5 ***
x0 = [x1_0 x2_0];

    % Solve the system

[t,soln] = ode45(@(t,x) rhs(t,x),tt,x0,options);

    % Plot solutions



plot(soln(:,1),soln(:,2));
xlabel('$x_1$','Interpreter','latex','FontSize',20);
ylabel('$x_2$','Interpreter','latex','FontSize',20);

hold off
    % Right hand side of the system differential equations
    % *** Your entry 6 ***  (enter your system here)

function [xp] = rhs(t,x)   % do not change this
    alpha = 1.5;
    beta = 1.1;
    gamma = 2.5;
    delta = 1.4;
    x1prime = -alpha.*x(1) + beta.*x(2).*x(1);
    x2prime = gamma.*x(2) - delta.*x(2).*x(1);
    xp = [x1prime ; x2prime];  % do not change this
end                            % do not change this
```

6.6 Code for Figure 6

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rudimentary script to solve an initial value problem consisting of
% the two example differential equations here on the interval a<=t<=b
%
% x1'(t) = alpha*x1+beta*x2,  x1(a) = x1_0
% x2'(t) = gamma*x1+delta*x2, x2(a) = x2_0
%
% Look for "*** Your entry i ***" where i=1,2,3,4,5,6 to make appropriate
% changes to solve your specific system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Housekeeping

clc ;       % clear the command window
clear;      % clear all variables
close all ; % close all figures

    % ODE45 options

options = odeset('AbsTol',1e-10,'RelTol',1e-7);

    % Independent variable

a = 0;      % left endpoint of t interval     *** Your entry 1 ***
b = 30;   % right endpoint of t interval    *** Your entry 2 ***
h = 0.01;   % stepsize                        *** Your entry 3 ***
tt = a:h:b;

    % Initial value of dependent variables

x1_0 = 0.5;    % initial value of x1    *** Your entry 4 ***
x2_0 = 1;    % initial value of x2    *** Your entry 5 ***
x0 = [x1_0 x2_0];

    % Solve the system

[t,soln] = ode45(@(t,x) rhs(t,x),tt,x0,options);

    % Plot solutions
hold on
figure(1);    % time series of x1(t)
plot(t,soln(:,1));
plot(t,soln(:,2));
xlabel('$t$','Interpreter','latex','FontSize',20);
ylabel('$x$','Interpreter','latex','FontSize',20);
legend('x_1', 'x_2')



hold off

    % Right hand side of the system differential equations
    % *** Your entry 6 ***  (enter your system here)

function [xp] = rhs(t,x)   % do not change this
    alpha = 1.5;
    beta = 1.1;
    gamma = 2.5;
```

```
    delta = 1.4;
    x1prime = -alpha.*x(1) + beta.*x(2).*x(1);
    x2prime = gamma.*x(2) - delta.*x(2).*x(1);
    xp = [x1prime ; x2prime];  % do not change this
end                            % do not change this
```

6.7 Code for Figure 7

```
% Step 1: Set the axis limits so that you plot the vector field over the
%         intervals x1min < x1 < x1max, x2min < x2 < x2max
    x1min = -1; x1max = 6; x2min = -1; x2max = 6;

% Step 2: pick step sizes for x1 and x2;
    x1step = 1; x2step = 1;

% generate mesh for plotting
    [x1, x2] = meshgrid(x1min:x1step:x1max, x2min:x2step:x2max);

% Step 3: define all needed parameter values
    a = 1.5;
    B = 1.1;
    y = 2.5;
    S = 1.4;
    k = 0.5;

% Step 4: define the system of equations you are using
    dx1 = -a.*x1 + B.*x1.*x2;
    dx2 = y.*(1-k.*x2).*x2 - S.*x1.*x2;

% normalize vectors (to help plotting)
    dx1 = dx1./sqrt(dx1.^2 + dx2.^2);
    dx2 = dx2./sqrt(dx1.^2 + dx2.^2);

% generate the vector field
    hold on
    quiver(x1, x2, dx1,dx2,'AutoScaleFactor',0.5)

% specify the plotting axes
    axis([x1min x1max x2min x2max])

% Step 5: label the axes, include a title
    xlabel('$x1$','Interpreter','latex')
    ylabel('$x2$','Interpreter','latex')
    title('Vector field','Interpreter','latex')
    x2 = linspace(-1,6,600);
    x = (y.*(1-k.*x2))./S;
    plot(x, x2, 'k');
    x3 = (a./B);
    x4 = linspace(-1,6,600);
    %plot(x4,x3, 'r');


    xline(0,'r');

    yline(0,'k');

    yline(a/B,'r');

    Z = plot(0,0,'g');
    Y = plot(0,2,'g');
    X = plot(0.568,(1.5/1.1), 'g');
    set(Z,'marker','o','markersize',25)
    set(X,'marker','.','markersize',70)
    set(Y,'marker','o','markersize',25)
```

```
    options = odeset('AbsTol',1e-10,'RelTol',1e-7);

    % Independent variable

a = 0;      % left endpoint of t interval     *** Your entry 1 ***
b = 30;   % right endpoint of t interval    *** Your entry 2 ***
h = 0.01;   % stepsize                        *** Your entry 3 ***
tt = a:h:b;

    % Initial value of dependent variables

x1_0 = 1;    % initial value of x1    *** Your entry 4 ***
x2_0 = 5;    % initial value of x2    *** Your entry 5 ***
x0 = [x1_0 x2_0];
x1_1 = 5;
x2_1 = 1;
x1 = [x1_1 x2_1];




    % Solve the system

[t,soln] = ode45(@(t,x) rhs(t,x),tt,x0,options);
[t,soln2] = ode45(@(t,x) rhs(t,x),tt,x1,options);

    % Plot solutions

hold on

   % phase portrait
plot(soln(:,1),soln(:,2));
xlabel('$x_1$','Interpreter','latex','FontSize',20);
ylabel('$x_2$','Interpreter','latex','FontSize',20);
plot(soln2(:,1),soln2(:,2));
    % Right hand side of the system differential equations
    % *** Your entry 6 ***  (enter your system here)
    legend('Vector Field','H-nullcline(1)','V-nullcline(1)','H-nullcline(2)','V-nullcline↙
(2)','Unstable Equilibrium Point(1)','Unstable Equilibrium Point(2)','Stable Equilibrium↙
Point(1)','Trajectory 1','Trajectory 2')
hold off
function [xp] = rhs(t,x)   % do not change this
    a = 1.5;
    B = 1.1;
    y = 2.5;
    S = 1.4;
    k = 0.5;
    x1prime = -a*x(1) + B*x(1).*x(2);
    x2prime = y*(1 - k*x(2)).*x(2) - S*x(1).*x(2);
    xp = [x1prime ; x2prime];  % do not change this
end                            % do not change this
```

6.8 Code for Figure 8

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rudimentary script to solve an initial value problem consisting of
% the two example differential equations here on the interval a<=t<=b
%
% x1'(t) = alpha*x1+beta*x2,  x1(a) = x1_0
% x2'(t) = gamma*x1+delta*x2, x2(a) = x2_0
%
% Look for "*** Your entry i ***" where i=1,2,3,4,5,6 to make appropriate
% changes to solve your specific system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Housekeeping

clc ;       % clear the command window
clear;      % clear all variables
close all ; % close all figures


    % ODE45 options

options = odeset('AbsTol',1e-10,'RelTol',1e-7);

    % Independent variable

a = 0;      % left endpoint of t interval     *** Your entry 1 ***
b = 30;   % right endpoint of t interval    *** Your entry 2 ***
h = 0.01;   % stepsize                        *** Your entry 3 ***
tt = a:h:b;

    % Initial value of dependent variables

x1_0 = 5;    % initial value of x1    *** Your entry 4 ***
x2_0 = 1;    % initial value of x2    *** Your entry 5 ***
x0 = [x1_0 x2_0];
x3_0 = 1;
x4_0 = 5;
x1 = [x3_0 x4_0];


    % Solve the system

[t,soln] = ode45(@(t,x) rhs(t,x),tt,x0,options);

    % Plot solutions

hold on
figure(1);    % time series of x1(t)
plot(t,soln(:,1));
xlabel('$t$','Interpreter','latex','FontSize',20);
plot(t,soln(:,2));
title('Plot of Component Curves(Initial Condition 1)','Interpreter','latex')
legend('x(1)t','x(2)t')
hold off

figure(2);    % phase portrait
plot(soln(:,1),soln(:,2));
xlabel('$x_1$','Interpreter','latex','FontSize',20);
ylabel('$x_2$','Interpreter','latex','FontSize',20)
```

```
    % Right hand side of the system differential equations
    % *** Your entry 6 ***  (enter your system here)

function [xp] = rhs(t,x)   % do not change this
    a = 1.5;
    B = 1.1;
    y = 2.5;
    S = 1.4;
    k = 0.5;
    x1prime = -a*x(1) + B*x(1).*x(2);
    x2prime = y*(1 - k*x(2)).*x(2) - S*x(1).*x(2);
    xp = [x1prime ; x2prime];  % do not change this
end                            % do not change this
```

6.9 Code for Figure 9

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rudimentary script to solve an initial value problem consisting of
% the two example differential equations here on the interval a<=t<=b
%
% x1'(t) = alpha*x1+beta*x2,  x1(a) = x1_0
% x2'(t) = gamma*x1+delta*x2, x2(a) = x2_0
%
% Look for "*** Your entry i ***" where i=1,2,3,4,5,6 to make appropriate
% changes to solve your specific system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Housekeeping

clc ;       % clear the command window
clear;      % clear all variables
close all ; % close all figures


    % ODE45 options

options = odeset('AbsTol',1e-10,'RelTol',1e-7);

    % Independent variable

a = 0;      % left endpoint of t interval     *** Your entry 1 ***
b = 30;   % right endpoint of t interval    *** Your entry 2 ***
h = 0.01;   % stepsize                        *** Your entry 3 ***
tt = a:h:b;

    % Initial value of dependent variables

x1_0 = 1;    % initial value of x1    *** Your entry 4 ***
x2_0 = 5;    % initial value of x2    *** Your entry 5 ***
x0 = [x1_0 x2_0];



    % Solve the system

[t,soln] = ode45(@(t,x) rhs(t,x),tt,x0,options);

    % Plot solutions

hold on
figure(1);    % time series of x1(t)
plot(t,soln(:,1));
xlabel('$t$','Interpreter','latex','FontSize',20);
plot(t,soln(:,2));
title('Plot of Component Curves(Initial Condition 2)','Interpreter','latex')
legend('x(1)t','x(2)t')
hold off

figure(2);    % phase portrait
plot(soln(:,1),soln(:,2));
xlabel('$x_1$','Interpreter','latex','FontSize',20);
ylabel('$x_2$','Interpreter','latex','FontSize',20)
    % Right hand side of the system differential equations
    % *** Your entry 6 ***  (enter your system here)
```

```
function [xp] = rhs(t,x)   % do not change this
    a = 1.5;
    B = 1.1;
    y = 2.5;
    S = 1.4;
    k = 0.5;
    x1prime = -a*x(1) + B*x(1).*x(2);
    x2prime = y*(1 - k*x(2)).*x(2) - S*x(1).*x(2);
    xp = [x1prime ; x2prime];  % do not change this
end                            % do not change this
```

7 Appendix C: Additional Figures

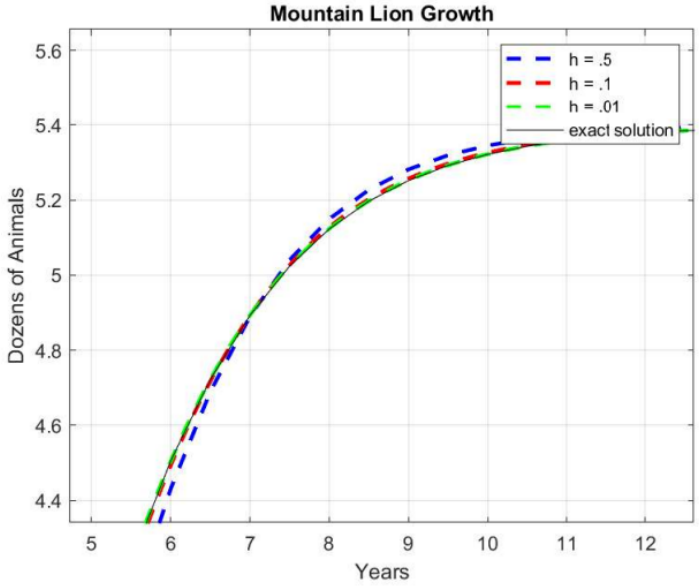

Figure 10: Mountain lion growth zoomed in near $t = 7$