A Mathematical Investigation of Populations and Predator-Prey Dynamics

Graham Miller Drake Nosker Ryan Senne

10/1/2021

1 Introduction

In the 1950's the Sierra Nevada mule deer population experienced an unprecedented depression that could not be adequately described by overpopulation of the native species as eventually the population dropped below the local carrying capacity. Interestingly enough, the mountain lion population of the Sierra Nevada was continuing to increase, possibly suggesting that these two populations may be interdependent. Using several population and predator-prey differential equation models, we can simulate effects certain environmental factors have on the populations of the local mule deer and mountain lion species.

2 A Logistical Model for Individual Populations

One way to model individual populations is to use a simple logistical model that accounts for environmental factors, but neglects the effects of other local fauna. In the Sierra Nevada, mountain lions have no natural predators and thus constraints to their population are a result on access to adequate food supplies. The local mule deer are one of the mountain lions main food source, but as we do not yet consider intra-species interactions we must account for their finite growth in our model with some other parameters. Examine the following linear, first-order, autonomous differential equation :

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

In this differential equation, the change in mountain lion population x (in dozens of mountain lions) per time t (years) is dependent on r, the growth rate of the population with units *lions/year*, and L (where L>0) represents the carrying capacity of the species. The carrying capacity is a measure of the maximum possible population size that is able to be sustained given the available resources in the environment, and is measured in AMU(animal unit equivalent). It is of mathematical note that since the differential equation that models the mountain lion population is autonomous, the population

change does not rely on the independent variable time t, but instead is entirely dependent on the prior years population. (1)

When further analyzing the differential equation, its found that the equilibrium solution is equal to the carrying capacity, $X = L$, and the non-equilibrium solution is represented by the equation:

$$x(t) = \frac{(\frac{x_0}{L-x_0})e^{rt}L}{1 + (\frac{x_0}{L-x_0})e^{rt}} \tag{2}$$

If we assume that $r = 0.65$, $L = 5.4$, and $x_0 = 6$ the solution of the initial value problem (IVP) becomes:

$$x(t) = \frac{-54e^{0.65t}}{1 - 10e^{0.65t}} \tag{3}$$

This equation will give us an exact solution curve for the IVP that we can compare to numerical approximations via Euler's method defined as:

$$f(x,t)_n = f(x,t)_{n-1} + hf(x,t)_{n-1} \tag{4}$$

where h is the step size at each iteration. Using Euler's method with varying step sizes h=0.5, 0.1, and 0.01 will result in three different numerical solution curves which can be plotted against the exact solution curve, shown in figure 1 below:

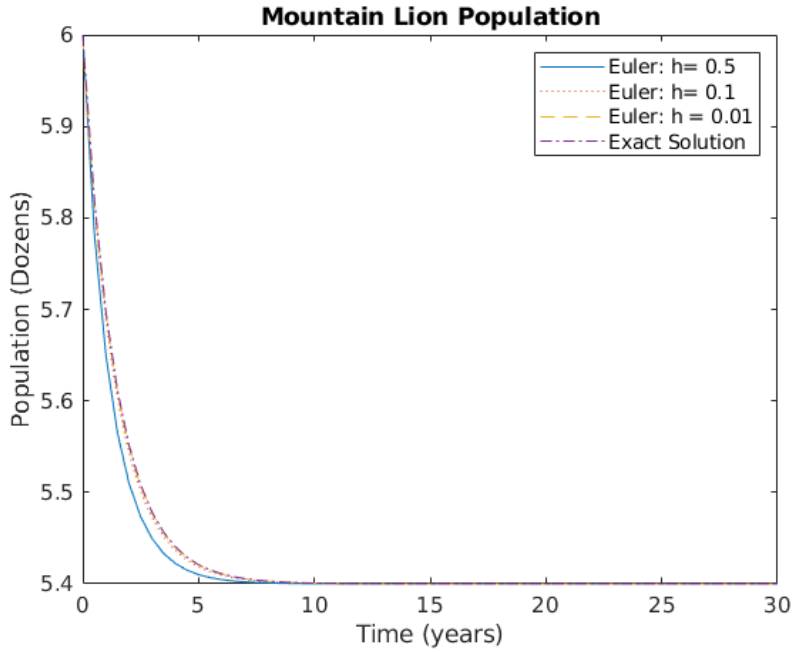

Figure 1: Simulation of Mountain Lion populations according to equation 1. Numerical solutions use step size (h=0.5, 0.1, and 0.01). Analytical solution plotted according to equation 2.

All solution curved devolve to the carrying capacity $L = 5.4$ (dozen).

When comparing the exact solution curve to the numerical solution curves derived via Euler's method, it can be observed that a smaller step size results in a numerical solution curve that more accurately matches the exact solution curve. A smaller step size means that a smaller $\Delta\ x$ range is used to calculate an approximate slope to the exact solution curve at each iteration of Euler's method. As a result of a smaller step size, more iterations of Euler's method need to be performed, and a higher resolution of individual slopes will be used to create the numerically derived solution curve.

The accuracy of each numerical solution curve when compared to the exact solution curve can be expressed through the equation for absolute error:

$$\text{Absolute Error} = |\text{Exact Solution - Approximate Solution}| \tag{5}$$

Plotting the results from applying the absolute error function to each numerical solution results in figure 2:

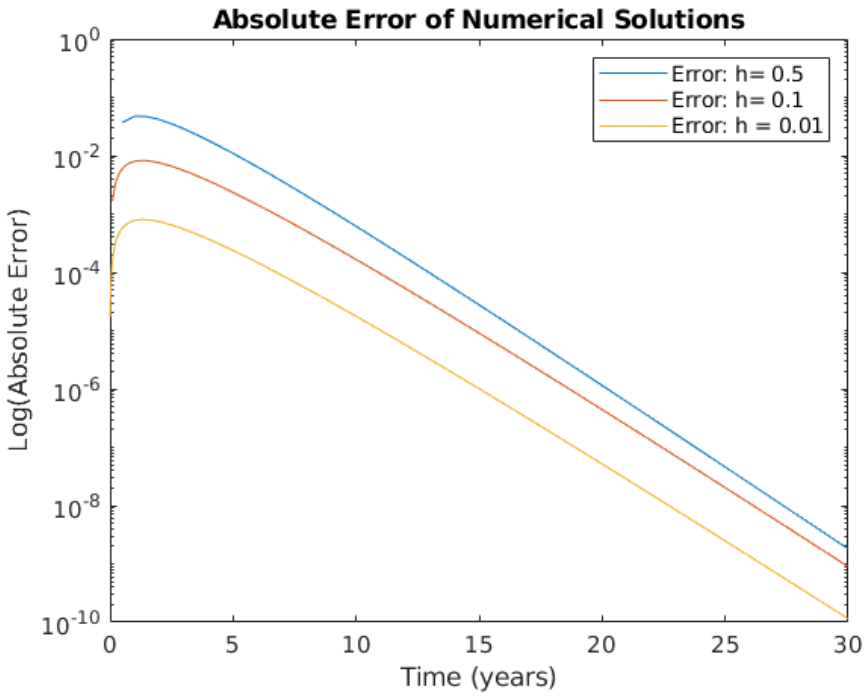

Figure 2: Absolute Error of Numerical Solution Curves. Absolute error defined according to Eq(5) Maximum error spikes at around the seventh iteration of Euler's method.

When using computational methods to approximate a solution, it is important to balance both accuracy and efficiency. Referring to Euler's method, accuracy and efficiency are inversely proportional and are a result of the step size. As step size decreases, accuracy increases and efficiency decreases, and as step size increases,accuracy decreases and efficiency increases. Keeping this in mind, a step size of 0.1 offers the best balance of precision to efficiency to find a numerical approximation to the exact

solution curve. A step size of 0.1 results in a substantially more accurate numerical estimate then the larger step size of 0.5, as shown in figures 1 and figure 2. Additionally, a step size of 0.1 requires one-tenth the total number of iterations required when compared to a step size of 0.01, resulting in a much greater efficiency.

The differential equation used to represent the mule deer population is similar to that of the mountain lion population, having parameters growth rate r and carrying capacity L. However, since the mule deer population is also directly impacted by predation from the mountain lions, it is sensible that the new differential equation should include a harvesting term *H(x)*. Thus, we can model the mule deer population$x$ (in dozens of deer) per time t time (years) through the linear, first order, autonomous differential equation:

$$\frac{dx}{dt} = r(1 - \frac{x}{L})x - H(x) \tag{6}$$

It should be noted that autonomous nature of this differential equation means that the population change of the mule deer is not dependent on time t, and is only dependent on the prior years population. The harvesting term *H(x)* can be further defined by parameters p and q, which represent how skilled the mountain lions are at catching the deer:

$$H(x) = \frac{px^2}{q + x^2} \tag{7}$$

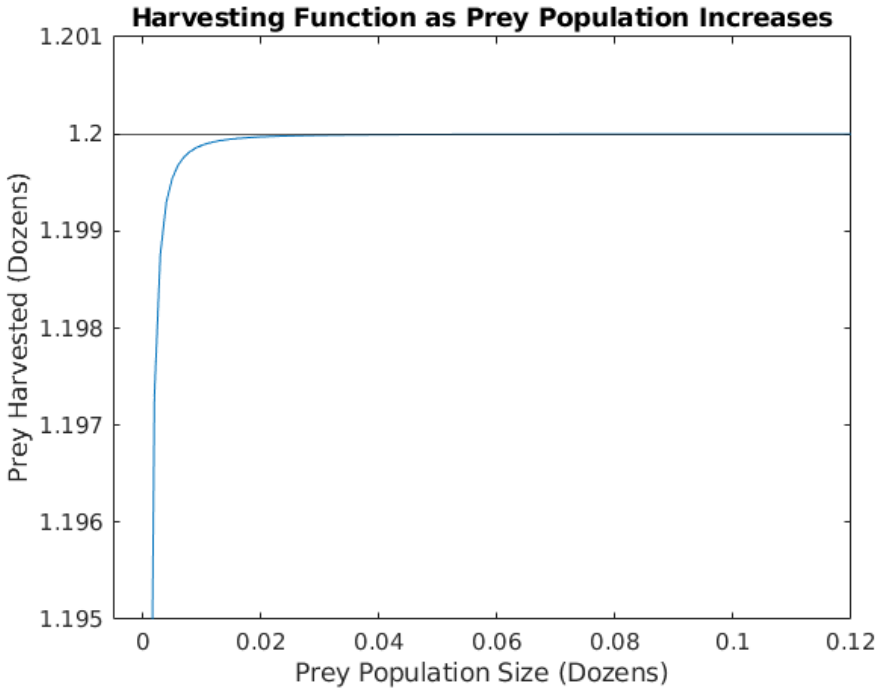

Figure 3: Harvesting Function *H(x)* behavior when p=1.2 and q=1.0. Harvesting

The graph of the harvesting function shows that as the population of deer becomes infinitely large, *H(x)* approaches p=1.2, the parameter which represents how skilled the mountain lions are at catching the deer. Intuitively this makes sense, since if there are an infinite number of deer to catch, then the mountain lion's skill at catching the deer is the only limiting factor on the total number of deer harvested. By similar logic, as the deer population approaches zero, the number of deer harvested approaches zero, since if there are no deer available for the mountain lions to catch, no deer will be harvested.

When applying initial conditions r=0.65, L=8.1, p=1.2 and q=1.0, it is found that the positive equilibrium solutions to the differential equation (rounded to 4 decimal spots) are x=0.8018, 1.8564, 5.4418

Varying the initial mule deer population size (in dozens of deer) to x_0=84, 24, 18, 6 and numerically solving each respective IVP using Euler's method with a step size of 0.1 will result in four different solution curves. Each solution curve, along with the direction field, and positive equilibrium solutions are shown in figure 4:

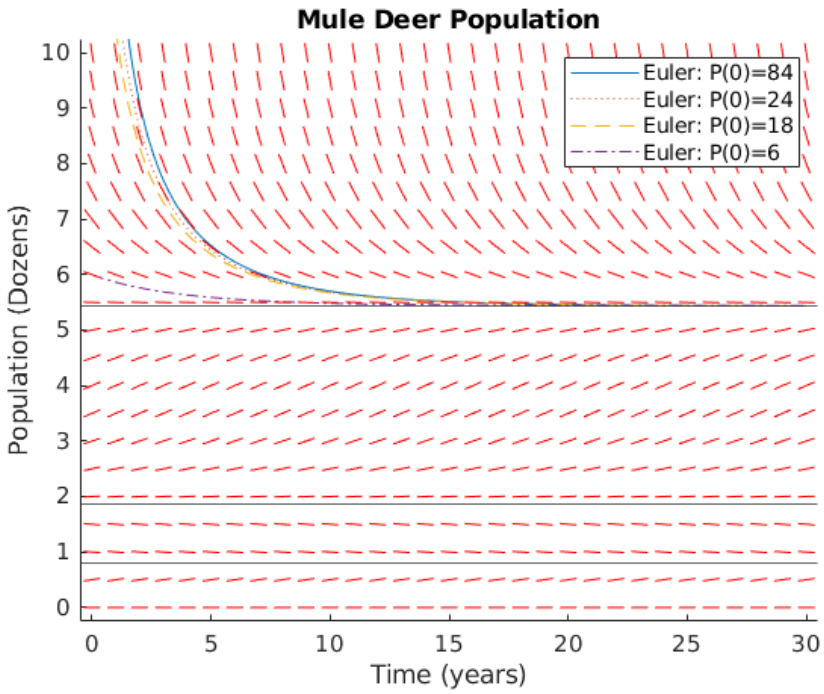

Figure 4: Simulation of Mule Deer Population at P(0), = 6, 18, 24, and 84. Numerical solutions use h=0.1 for step size. Equilibrium solutions present at p=0.8018, 1.8564, and 5.4418.

In all four initial population sizes, the deer population experiences an immediate decrease in size. The greater the initial population size is, the faster the initial deer population will decline. This can be explained by the reasoning that when there are more mule deer in the environment, it is easier for the mountain lions to prey on the deer, resulting in deer being harvested at a faster rate.

Moreover, the greater the deer population exceeds the carrying capacity of the environment, the greater the competition for resources will be, and faster the deer population will decrease. It is clear when looking at the slope field that any initial population size between $(1.8564, 5.4418) \cup (5.4418, \infty)$ stabilizes to the equilibrium solution x=5.4418. Examining the graph, we see that this is true for all four solution curves, which approach the equilibrium solution of 5.4418 dozens of deer.

3 Systems for Interdependent Population Dynamics

In the previous model the logistic equation was used to separately study the population and growth of both the mountain lion, and mule deer. Unique terms, like the Harvesting equation, were included to account for effects on those populations independent of one another. In these models the interactions between the two species are not represented separately through variables, but are studied more directly as a systems of equations.

3.1 The Lotka-Volterra System

Since the mule deer and mountain lion populations are known to have a prey and predator interaction, one technique that can be used to model this interaction is the Lotka-Volterra System. Using a system of the non-autonomous, first order, nonlinear, differential equations, each respective population can be represented as:

$$\frac{dx_1}{dt} = -\alpha x_1 + \beta x_1 x_2 \tag{8}$$

$$\frac{dx_2}{dt} = \gamma x_2 - \delta x_1 x_2 \tag{9}$$

Where x_1=predator population, x_2=prey population, α=predator mortality rate, β=predator attack rate, γ=prey growth rate, δ=prey mortality rate. Upon initial investigation it is advantageous to analytically solve for equilibrium values. In order to find the equilibrium points the equations must be solved for zero growth in terms of x1 and x2. These solutions are recognized as being vertical and horizontal nullclines, corresponding to the variable solution.

$$\text{V-nullclines: } x_1 = 0 \; x_2 = \frac{\alpha}{\beta}$$

$$\text{H-nullclines: } x_1 = \frac{\gamma}{\delta} \; x_2 = 0$$

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

In order to make the system palpable and find quantitative solutions, parameter values are assigned such that α=1.5, β=1.1, γ=2.5, and δ=1.4 . The system of equations are then able to be depicted as a vector field as well. The corresponding vertical and horizontal nullclines are also graphed.

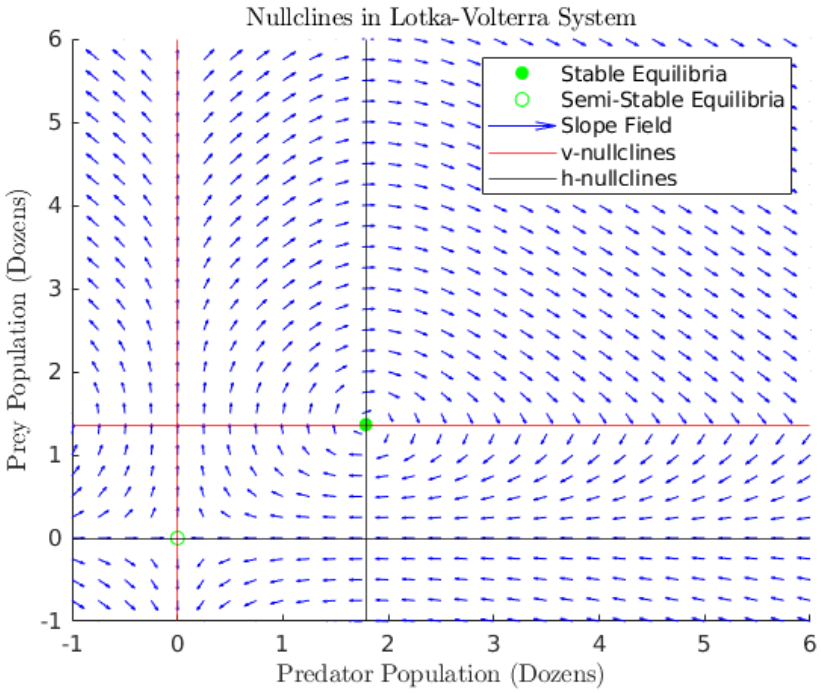

Figure 5: Phase portrait of Lotka-Volterra system. Vertical nullclines (red) and horizontal nullclines (black) are graphed. There is a semi-stable saddle point at (0,0) and a stable spiral at approximately (1.798, 1.36).

Analyzing the graph, it is shown that the equilibrium points are located at the intersection of the V-nullclines and H-Nullclines. The vector field at the intersection of the x_2 V-nullcline and the x_1 H-nullcline exhibits an inward spiral pattern. This is suggests that it is a stable equilibrium point. The vector field at the intersection of the x_1 V-nullcline and x_2 H-nullcline exhibits a more complex behavior. Along the x_2 H-nullcline, vector field arrows point towards the equilibrium point from both sides of the intersection, while the vector field arrows on the x_1 V-nullcline point away from the equilibrium point of either side of the intersection. This behavior is indicative of a semi-stable equilibrium point.

To further graphically analyse the Lotka-Volterra predator-prey system, an inital condition of $X_1(0) = 0.5$ and $X_2(0) = 1.0$ are applied to the differential equations. Over time interval $t \in [0, 30]$, the following solution curve is generated:

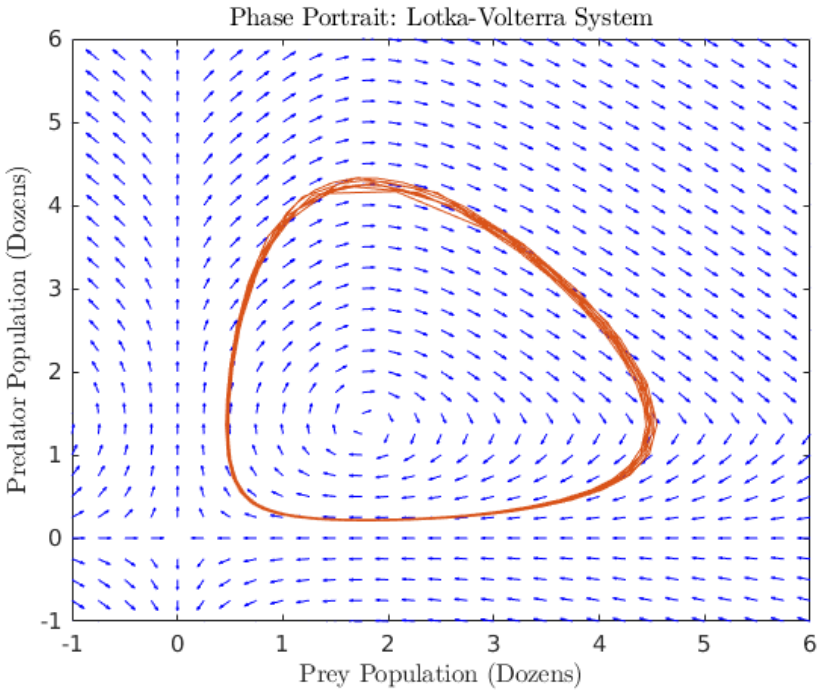

Figure 6: Phase portrait of Lotka-Volterra system where $\alpha = 1.5, \beta = 1.1, \gamma = 2.4, \delta = 1.4$. Solution curve for $t \in (0, 30)$. Particular solution represents a limit cycle.

Based on the equilibrium stability and directional vector field discussed above, the solution curve behaves as expected. The trajectory plot verifies that $x_1 = L$, x2= α/β, is a stable equilibrium. This conclusion can be made because the solution curve forms a closed loop around the stable equilibrium point. The solution curve would travel towards the semi-stable equilibrium point only if the prey population value went to zero, and because the predator population depends on the prey population, the predator population would would also go to zero. The cyclic nature of the solution curve can be interpreted as meaning when the predator population decreases, the prey population increases rapidly, and as a result, there is more food for the predators to consume, so the predator population then increases. However, now that there are more predators, the prey population declines, and causing there to be less food for the predators, and the predator population declines, thus starting the cycle over.

Using the same initial conditions, the population of x_1 and x_2 can then be plotted against one another over the interval $t \in (0, 30)$., resulting in the figure below:

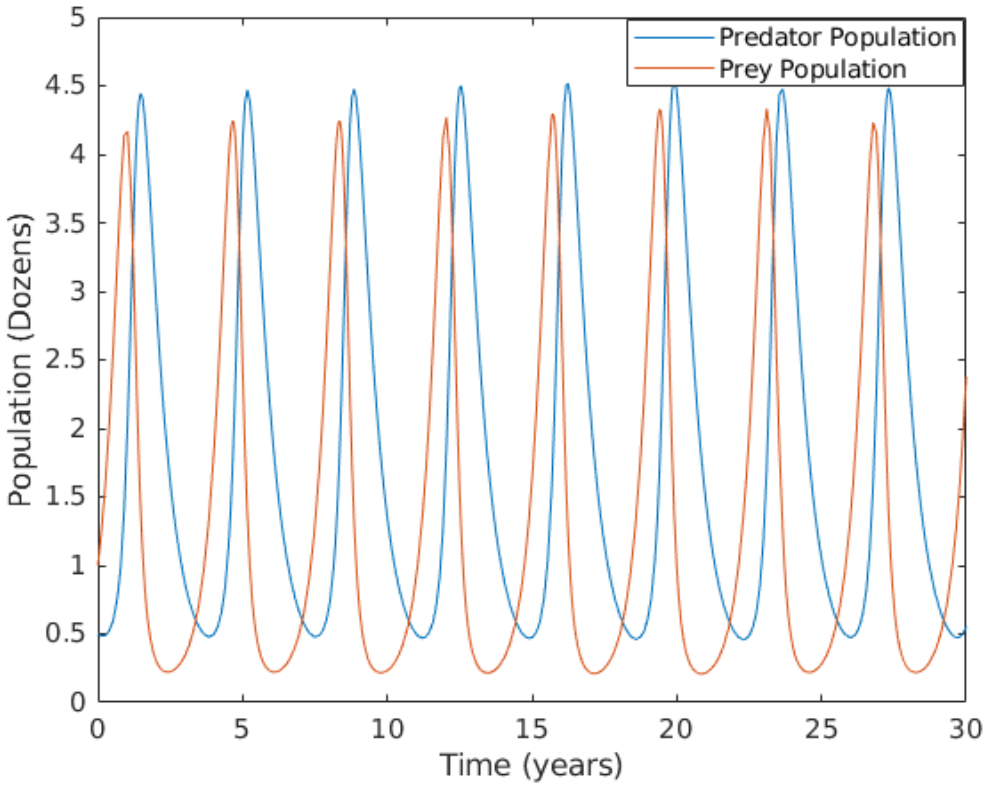

Figure 7: Graph of component time series for predator and prey populations for time $t \in (0, 30)$.. Components exhibit strong periodic nature.

Figure 7 shows the mountain lion and mule deer populations are slightly out of phase with one another, with the peaks and troughs of the predator population x_1 being occurring slightly after those of prey population x_2. It is also of importance to recognize that over the majority of the graph, both populations are increasing and/or decreasing at the same time, and it is only over small intervals where the prey population increases while the predator population decreases, or the prey population decreases while the predator population increases. The graph also shows a continuous oscillating nature to both species total populations. Taking all of these details into account, the graph can then be interpreted to describe the interdependent nature each population has on one another. Starting at t=0, the predator and prey population begin to increase, the predator population reaches a population size at which the prey population cannot sustain its population size anymore, and it begins to decrease. As a result of the decrease in prey population, the predator population then decreases shortly after. In turn, there are less predators to eat the prey, so the prey population begins to increase again before the predator population has had time to recover. Then as a result of the prey population increasing again, the predators have more prey available to feed on, and the predators population begins to increase again, thus starting the cycle over.

3.2 The Logistic Predator-Prey Equations

In the Logistic Predator-Prey model, the predator/prey interactions between the species are accounted for, as well as constraining environmental factor exhibited on the prey population, such as finite food, resulting in a more accurate model. The resulting Logistic Predator-Prey system of differential equations are:

$$\frac{dx_1}{dt} = -\alpha x_1 + \beta x_1 x_2 \tag{10}$$

$$\frac{dx_2}{dt} = \gamma(1 - kx_2)x_2 - \delta x_1 x_2 \tag{11}$$

Where k represents the limiting environmental factors on the prey population, and all other variables hold the same meaning as they did with the lotka-volterra differential equations. Likewise to previous models, it is advantageous to analytically find equilibrium solutions of the differential equations. In order to find the equilibrium points the equations must be solved for zero growth in terms of x1 and x2. These solutions are recognized as being vertical and horizontal nullclines, corresponding to the variable solution:

$$\text{V-nullclines: } x_1 = 0 \; x_2 = \frac{\alpha}{\beta}$$

$$\text{H-nullclines: } x_1 = \frac{\gamma - k\gamma(\alpha/\beta)}{\delta} \; x_2 = 0$$

$$\text{Equilibrium Solutions: } (0,0) \text{ and } (\frac{\alpha}{\beta}, \frac{\gamma - k\gamma(\alpha/\beta)}{\delta})$$

In order to make the system palpable and find quantitative solutions, parameter values are assigned such that $\alpha = 1.5$, $\beta = 1.1$, $\gamma = 2.5$, $\delta = 1.4$ and $\kappa = 0.5$. The system of equations are then able to be depicted as a vector field, along with the corresponding vertical and horizontal nullclines. Applying initial conditions $(x_1(0), x_2(0)) = (5, 1)$ and $(x_1(0), x_2(0)) = (1, 5)$ over the time interval $t \in [0, 30]$ will result in two unique trajectories.

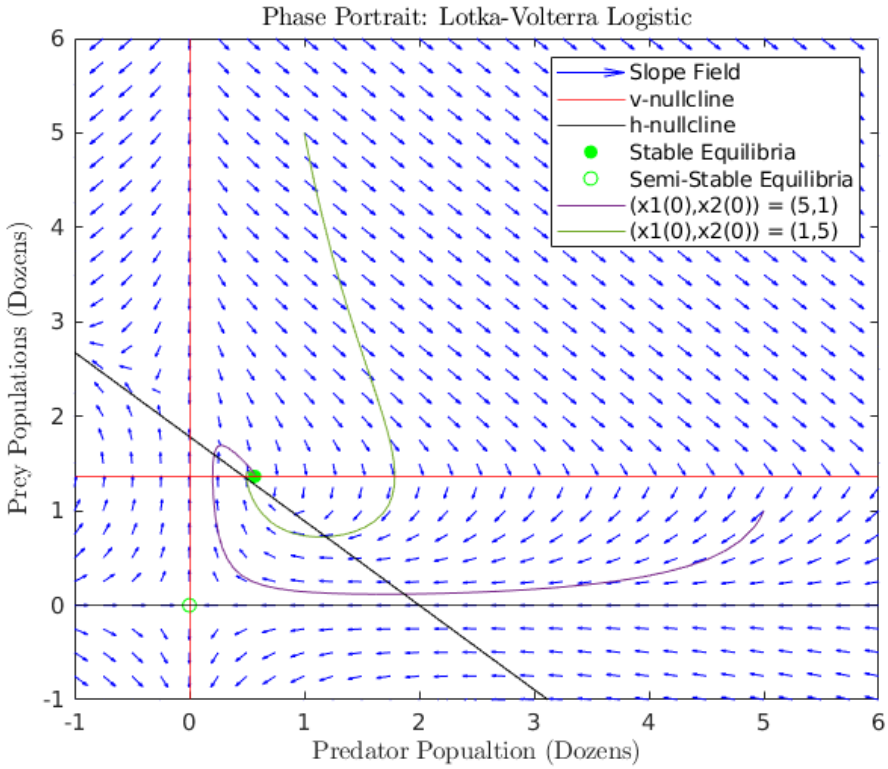

Figure 8: Phase Portrait of Logistic-Lotka-Volterra System. Two IVP are graphed where (x1(0),x2(0) = (5,1) or (1,5). Vertical nullclines and horizontal nullclines are plotted in red and black respectively. Both solution curves evolve to a asymptotically stable node $(\alpha/\beta, \frac{\gamma - k\gamma(\alpha/\beta)}{\delta})$.

Analyzing the graph, it is shown that the equilibrium points are located at the intersection of the V-nullclines and H-nullclines. The vector field at the intersection of the x_2 V-nullcline and the x_1 H-nullcline exhibits behavior of an asymptotically stable node. This suggests that it is a stable equilibrium point. The vector field at the intersection of the x_1 V-nullcline and x_2 H-nullcline exhibits nearly identical behavior to the equilibrium point at the origin for the Lotka-Volterra system, and thus for the same reasoning, this equilibrium point is classified as semi-stable. The trajectory plots indicate what path the solution curves take starting with initial condition values for the population. It is also of important note that both solution curves, despite having a different initial value, end up converging to the stable equilibrium point.

To better see how the population of the Logistic Predator-Prey equations interact with one another, the populations x_1 and x_2 can be plotted against one another over the interval $t \in [0, 30]$

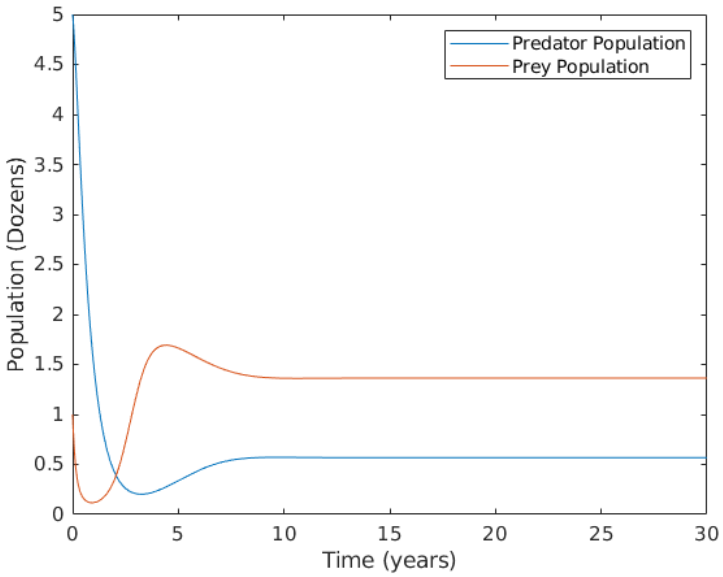

Figure 9: Component graph for predator and prey populations where (x1(0),x2(0)= (1, 5). Both curves exhibit asymptotic behavior. They approach ≈ 1.3636 and 0.56822 respectively. The exact asymptotes are at α/β and at $\frac{\gamma - k\gamma(\alpha/\beta)}{\delta}$.

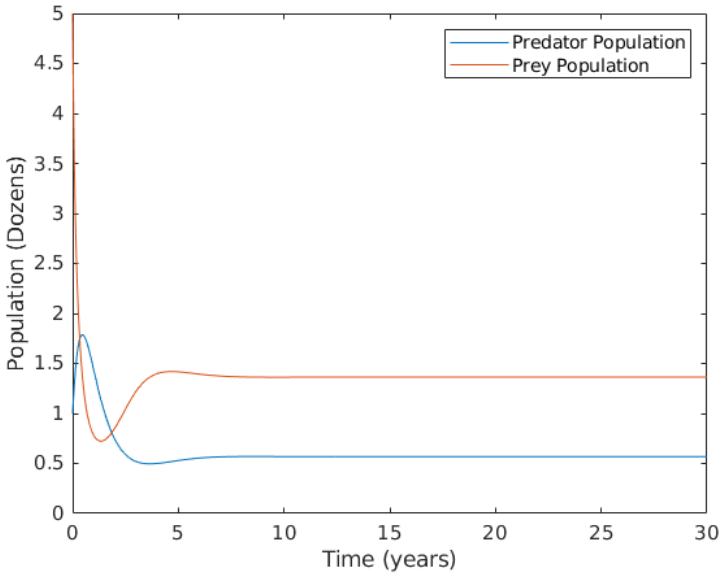

Figure 10: Component graph for predator and prey populations where (x1(0),x2(0)= (5, 1). Both curves exhibit asymptotic behavior. They approach ≈ 1.3636 and 0.56822 respectively. The exact asymptotes are at α/β and at $\frac{\gamma - k\gamma(\alpha/\beta)}{\delta}$.

Referring to figure 9, both the mountain lion and mule deer populations experience a sharp drop in population. Referring to figure 10, the mountain lion population experience a sharp initial drop in population, while the mule deer population experiences an initial increase in population size. Neither population in either graph experiences any continuous periodic behavior, and rather the mountain lion and mule deer population exhibit strong asymptotic behavior, with the mountain lion population approaching 0.56822 (dozens of mountain lion) and the mule deer population approaching 0.56822 (dozens of mule deer), which are representative of the equilibrium solutions for the given parameters.

4 Discussion

We have discussed the application of three types of models: the logistic model, the Lotka-Volterra system, and a combined Logistic Lotka-Volterra system. These models have their implicit advantages and disadvantages. Particularly of interest is the logistic and Lotka-Volterra models. The logistic model relies on a series of assumptions that allow it to have an incredible simplicity, and for this reason, it can be easily solved analytically, and thus exact solutions can be derived. However, its simplicity is also a weakness, because it relies on so many assumptions: in the mountain lion case that there is no dependence on the population size of the deer populations, and that there is really only some carrying capacity for the population to converge to, there is likely only very specific circumstances in which this model would really hold true. This isn't to say it is worthless, only that its in many ways oversimplified. Contrasting this to the Lotka-Volterra system, which is a system of equations that relates the deer and mountain lions. This model then becomes significantly more sophisticated, in that it has four parameters to consider in comparison to the logistic model which only relies on the growth rate of the species, r, the carrying capacity, L, and the harvesting equation in the case of mule-deer populations. This has two effects, there is possibly more control when modeling the population curves, but there is also need for more data to accurately set these parameters. This also makes analytical solutions to be harder to find in the Lotka-Volterra system due to the higher complexity of the system. However, due to the rise of incredibly powerful computers, and several useful numerical approximations these concerns could likely be minimized with intelligent model design.

One possible way to make these more accurate is to add in more differential equations to make a more complex system. For example, one assumption of both models is that there is some static, unchanging, growth rate for the species. This is likely not realizable in natural circumstances and could therefore be a reason for there to be errors in the simulations. If some equation could be derived that acutely models growth rates of the species in times, this could eliminate this assumption. Another error in these models likely arises from other assumptions for example, there is no term that describes the natural hunger rate for our predators. It is unlikely that predators would hunt more than necessary because this represents an evolutionary disadvantage, as hunting has a negative cost associated to it. Thus if one could model, the willingness to hunt so to speak, that was dependent on the amount of deer harvested in some time point, then this could also eliminate this source of

error. One issue however, with having fewer assumptions, is that without assumptions our system can become arbitrarily complex, and there is likely a finitely large amount of parameters that could dictate the accuracy of our model and so it would be pertinent to quantify the efficiency change associated with adding complexity.

We've effectively shown the differences between the logistic, Lotka-Volterra, and Logistic-Lotka-Volterra system numerically, quantitatively, and numerically. None of these models are perfect, but they are incredibly powerful descriptions of reality that will have more or less accuracy dependent on the unique circumstances. It is easy then to see why the study of differential equations has such a strong influence on ecological studies.

5 Appendix

5.1 Question set A:

finding equilibrium solutions to Eq(1):

$$\frac{dx}{dt} = r(1 - \frac{x}{L})x$$
$$0 = rx - \frac{xr}{L}x$$
$$\frac{x^2 r}{L} = rx$$
$$x^2 = Lx$$
$$x = L \; x = 0$$

finding non-equilibrium solutions to Eq(1):

$$\frac{dx}{dt} = r(1 - \frac{x}{L})x$$
$$\frac{1}{(1 - \frac{x}{L})x}dx = rdt$$
$$\int \frac{1}{x}dx + \int \frac{1}{L(1 - \frac{x}{L})}dx = \int rdt$$
$$ln|\frac{x}{L - x}| = rt + C$$
$$\frac{x}{L - x} = Ce^{rt}$$
$$x = Ce^{rt}L - Ce^{rt}x$$
$$x = \frac{Ce^{rt}L}{1 + Ce^{rt}}$$
$$x_0 = \frac{CL}{1 + C}$$
$$C = \frac{x_0}{L - x_0}$$
$$x = \frac{(\frac{x_0}{L - x_0})e^{rt}L}{1 + (\frac{x_0}{L - x_0})e^{rt}}$$

5.2 Question set B:

1) Finding V-nullclines

$$\frac{dx_1}{dt} = -\alpha x_1 + \beta x_1 x_2$$
$$0 = x_1(-\alpha + \beta x_2)$$
$$x_1 = 0 \; x_2 = \frac{\alpha}{\beta}$$

2) Finding H-nullclines

$$\frac{dx_2}{dt} = \gamma x_2 - \delta x_1 x_2$$
$$0 = x_2(\gamma - \delta x_1)$$
$$x_2 = 0 \; x_1 = \frac{\gamma}{\delta}$$

5.3 Question set C:

1) Finding V-nullclines

$$\frac{dx_1}{dt} = -\alpha x_1 + \beta x_1 x_2$$
$$0 = x_1(-\alpha + \beta x_2)$$
$$x_1 = 0 \; x_2 = \frac{\alpha}{\beta}$$

2) Finding H-nullclines

$$\frac{dx_2}{dt} = \gamma(1 - kx_2)x_2 - \delta x_1 x_2$$
$$0 = x_2(\gamma(1 - kx_2) - \delta x_1)$$
$$x_2 = 0 \; x_1 = \frac{\gamma(1 - kx_2)}{\delta}$$

References

[Jer18] Beverly H. West Jerry Farlow, James E. Hall, Jean Marie McDill. *No Title*. Pearson, New York , NY, 2 edition, 2018.

[M.B] S. Harrell M.Beals, L. Gross. PREDATOR-PREY DYNAMICS.

[Ser] U.S. Forest Service. Mountain Lions vs. Deer.

[Tse] The Predator-Prey Equations. Technical report.

5.4 Matlab Code

```
% function handle for logisticPop.m
f = @logisticPop;

% pass logisticPop to eulersMethod with step size 0.5, 0.1, and 0.01
popLow = eulersMethod(f, 0.5, 0, 30, 6);
popMed = eulersMethod(f, 0.1, 0, 30, 6);
popHigh = eulersMethod(f, 0.01, 0, 30, 6);

% calculate analytical solution
actual_population = arrayfun(@ivp, linspace(0,30,61));

% calcualte errors for the three step sizes used to calculate this problem
errorpopLow = absoluteError(popLow(2, :), actual_population);
errorpopMed = absoluteError(popMed(2, :), arrayfun(@ivp, ...
    linspace(0,30,301)));
errorpopHigh = absoluteError(popHigh(2, :), arrayfun(@ivp, ...
    linspace(0,30,3001)));

% plot figure that models mountain lion population
figure
plot(popLow(1, :), popLow(2, :), popMed(1, :), popMed(2, :), ':', ...
    popHigh(1, :), popHigh(2, :), '--', linspace(0,30,61), ...
    actual_population, '-.')
title('Mountain Lion Population')
xlabel('Time (years)')
ylabel('Population (Dozens)')
legend('Euler: h= 0.5', 'Euler: h= 0.1', 'Euler: h = 0.01', ...
    'Exact Solution')

%plot figure that puts the abolsute error of the numerical solutions on
%the y-axis
figure
semilogy(linspace(0,30,61), errorpopLow, linspace(0,30,301), errorpopMed, ...
    linspace(0,30,3001), errorpopHigh)
title('Absolute Error of Numerical Solutions')
xlabel('Time (years)')
ylabel('Log(Absolute Error)')
legend('Error: h= 0.5', 'Error: h= 0.1', 'Error: h = 0.01')

%function handle for the logistic harvesting equation
f2 = @logisticHarvesting;

% find the threee equilibrium solutions
z0 = fzero(f2, 0.8);
z1 = fzero(f2, 1.85);
z2 = fzero(f2, 5.44);

% pass logistic harvestignng to eulers method with four different population
% sizes
pop84 = eulersMethod(f2, 0.1, 0, 30, 84);
pop24 = eulersMethod(f2, 0.1, 0, 30, 24);
pop18 = eulersMethod(f2, 0.1, 0, 30, 18);
```

```
pop6 = eulersMethod(f2, 0.1, 0, 30, 6);

% plot figure that has the directional field, the four solutions to the
% logistic harvesting function, and the equilibirum solutions
figure
hold on
plot(pop84(1, :), pop84(2, :), pop24(1, :), pop24(2, :), ':', pop18(1, :), ...
    pop18(2, :), '--', pop6(1, :), pop6(2, :), '-.')
dirfield(f2, 0:1:30, 0:0.5:10, 'Mule Deer Population')
yline(z0)
yline(z1)
yline(z2)
xlabel('Time (years)')
ylabel('Population')
legend('Euler: h= 0.5', 'Euler: h= 0.1', 'Euler: h = 0.01', ...
    'Actual Solution')

% function handle for harvesting equation
h = @harvesting;

% simulate harvesting equation as population grows exponentially large
harvesting_simulation = arrayfun(h, linspace(1, 10000, 1000));

% plot simualtion of harvesting function and p value
figure
plot(linspace(0, 1.0, 1000), harvesting_simulation);
yline(1.2)
xlabel('Prey Population Size (Dozens)')
ylabel('Prey Harvested (Dozens)')
title('Harvesting Function as Prey Population Increases')
axis([-0.005 0.12 1.195 1.201])
```

plot equilibrium solutions for lotka volterra system in question 3.2.2

```
figure
hold on
equX = [(2.5/1.4)];
equY = [(1.5/1.1)];
scatter(equX, equY,'g', 'filled')
scatter(0,0, 'g')
flow
xline(0, 'red')
yline(0, 'black')
yline((1.5/1.1), 'red')
xline((2.5/1.4), 'black')

% fucntion handle for lotkla volterra
f3 = @lotkaVolterra;

% pass lotka volterra to ODE45, varibale is a 2 by x array with time series
% and popualtion information
[t, P] = ode45(f3, [0,30], [0.5, 1]);

% plot phase portrait
```

```
figure
flow
hold on
plot(P(:,1), P(:,2))

% plot the components in time
figure
plot(t, P(:,1), t, P(:,2));
xlabel('Time (years)')
ylabel('Population')
```

lotka volterra logistic function handle

```
f4 = @lotkavolterraLogistic;

% pass lotka volterra logisic equation to ode45 with two sets of parameters
[t1, P1] = ode45(f4, [0,30], [5,1]);
[t2, P2] = ode45(f4, [0,30], [1,5]);

% plot the phase portrait of the lotka volterra logistic system with
% horizontal and vertical nullclines on a slope field
figure
flow
hold on
xline(0, 'black')
yline(0,'red')
yline((1.5/1.1), 'black');
legend('Slope Field', 'h-nullcline', 'v-nullcline')
% anonymous function for one of the clines
f5 = @(x2) ((2.5-(2.5*0.5*x2))/1.4);
hnull = arrayfun(f5, linspace(-1, 6, 100));
plot(P1(:,1), P1(:,2), P2(:,1), P2(:,2), linspace(-1, 6, 100), hnull, 'red');


% plot components in time
figure
plot(t1, P1(:,1), t1, P1(:,2));
xlabel('Time (years)')
ylabel('Population')
legend("Population 1", "Population 2")

% plot components in time for second set of initial conditions
figure
plot(t2, P2(:,1), t2, P2(:,2))
xlabel('Time (years)')
ylabel('Population')
legend("Population 1", "Population 2")
```

Published with MATLAB® R2021b

```
function dydt = eulersMethod(f, h, a, b, y0)
% implementation of eulers method
% defined as f(x,t)_n = f(x,t)_n-1 + h*f(x,t)_n-1
% input:
% f: a differential equation fucntion handle
% h: step size
% a: left time point
% b: right time point
% y0: initial condition
n_steps = (b-a)/h;
y = zeros(n_steps+1, 1);
x = (a:h:b);
y(1) = y0;
    for i=1:n_steps
        y(i+1)=y(i)+h*f(y(i));
    end
dydt = [x; y'];

end
```

Published with MATLAB® R2021b

```
function dpdt = logisticPop(x)
% this function describes a logistic model of population growth that will
% be passed iteratively to a eulersMethod.m
% input:
% x : population size at some time point t
r = 0.65;
L = 5.4;
dpdt = r*(1-(x/L))*x;
end
```

Published with MATLAB® R2021b

```
function population = ivp(t)
% exact solution to IVP
% input
% t: time point
     population = (-54)*exp(0.65*t)/(1-10*exp(0.65*t));
end
```

Published with MATLAB® R2021b

```
function dpdt = logisticPop2(x)
% this function describes a logistic model of population growth that will
% be passed iteratively to a eulersMethod.m
% input:
% x : population size at some time point t
r = 0.65;
L = 8.1;
dpdt = r*(1-(x/L))*x;
end
```

Published with MATLAB® R2021b

```
function harvested = harvesting(x)
% harvesting function
% input
% x: population size at time point t
p=1.2;
q=1;
harvested = (p*(x^2))/(q+(x^2));
end
```

Published with MATLAB® R2021b

```
function dpdt = logisticHarvesting(p)
% combination of logistic equation2 and the harvesting function
% see logisticPop2.m and ahrvesting.m for further explanation
% p: population
    dpdt = logisticPop2(p) - harvesting(p);
end
```

Published with MATLAB® R2021b

```
function error = absoluteError(observed, actual)
% absolute error function
% input:
% observed: observed value
% actual: actual solution
    error = abs(actual - observed);
end
```

Published with MATLAB® R2021b

```
function dxdt= lotkaVolterra(~, p)
% this is an implementation of a lotka-volterra model of
    % differential equations
    %input:
    % ~: dummy variable for time that is needed to pass to ODE45
    % P: population row vector of size 2X1
    a=1.5;
    b=1.1;
    y=2.5;
    d=1.4;
    dxdt = zeros(2,1);
    dxdt(1) = (-a*p(1))+(b*p(1)*p(2));
    dxdt(2) = (y*p(2))-(d*p(1)*p(2));
end
```

Published with MATLAB® R2021b

```
function dxdt = lotkavolterraLogistic(~, P)
    % this is an implementation of a lotka-volterra logistic model of
    % differential equations
    %input:
    % ~: dummy variable for time that is needed to pass to ODE45
    % P: population row vector of size 2X1
    a = 1.5;
    b = 1.1;
    y = 2.5;
    d = 1.4;
    k = 0.5;
    dxdt = zeros(2,1);
    dxdt(1) = (-a*P(1))+(b*P(1)*P(2));
    dxdt(2) = (y*(1-k*P(2))*P(2))-(d*P(1)*P(2));
end
```

Published with MATLAB® R2021b

```
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
%   dirfield(f, -1:.2:3, -2:.2:2)

[tm,ym]=meshgrid(tval,yval);
dt = tval(2) - tval(1);
dy = yval(2) - yval(1);
fv = f;
if isa(f,'function_handle')
    fv = arrayfun(fv, ym);
end
yp=fv;
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');
```

Published with MATLAB® R2021b

```
% 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 = 0.25; x2step = 0.25;

% 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;
    d = 1.4;
    k = 0.5;

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

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


% generate the vector field
    quiver(x1, x2, dx1, dx2, 'blue', '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')
```

Published with MATLAB® R2021b