Interactive Ecological Population Models Website: A Master's Thesis Project

A Thesis Presented to

The Faculty of the Mathematics Program

California State University Channel Islands

In (Partial) Fulfillment

of the Requirements for the Degree

Master of Science

by

Elisabeth Perkins

December, 2012

Signature page for the Masters in Mathematics Thesis of Elisabeth Perkins

APPROVED FOR THE MATHEMATICS PROGRAM

Geoffrey Buhl 12/18/2012

Dr. Geoffrey Buhl, Thesis Advisor Date

Sean Anderson 12/18/12

Dr. Sean Anderson, Thesis Committee Date

APPROVED FOR THE UNIVERSITY

12-20-12

Dr. Gary A. Berg, AVP Extended University Date

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author s| retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author s| or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name|s| as the author s| or owner s| of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Interactive Ecological Population Models Website

Title of Item

Mathematics Thesis, Applets, Population Dynamics

3 to 5 keywords or phrases to describe the item

Elisabeth Perkins

Author s| Name (Print)

Author|s| Signature

01/02/2013

Date

Acknowledgements

I would like to thank my advisors, Dr. Geoff Buhl and Dr. Sean Anderson, for all their help and suggestions. I have to thank my parents, Rick and Rebecca Perkins, for contributing their wildlife photographs, assisting me in designing the website, and proofreading. My sisters, Samantha Perkins and Patricia Perkins, gets thanks for reading over my material. I also have to thank Brandon Ausmus for his unending support and encouragement.

Abstract

Interactive Ecological Population Models Website: A Master's Thesis Project

by Elisabeth Perkins

For this Master's Thesis Project, I created a website that educates people about nine different population models. I wrote several Java applets that allow users to explore both single population and multiple population models. Additionally, I explained each model from a combined ecological and mathematical perspective, and gave examples in the text that pair with examples in the programs. Most of the models use differential equations. This website explains how differential equations work, along with why and how the differential equations are created for the various models. In order to create graphs of the populations, the fourth order Runge-Kutta method was used to find approximations of the solutions of the systems of differential equations. One

model incorporates randomness through stochastic differential equations, and uses a discrete approximation technique to generate the data points.

Contents

1. Introduction

A population model is a mathematical model that describes the size of a population at any time. We use population models to help us understand population dynamics and to predict the future of populations. We can also use models to help us understand the impact we have on animal populations, to try to keep populations from going extinct, and to find the maximum sustainable yield of a population that we are harvesting.

There are many population models that can be used for various purposes and situations. The exponential model, for example, can be a good model to use for short-term prediction of a population with unlimited resources. It is not a good model, however, for long-term predictions, because resources will eventually run out.

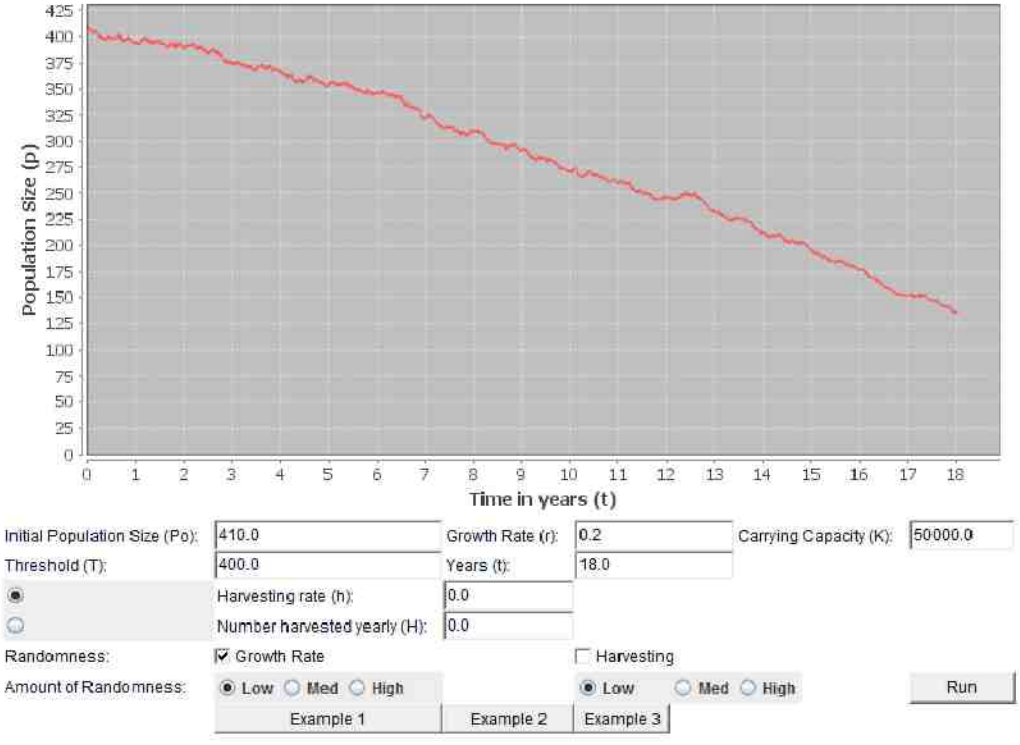
Students are typically exposed to basic ecological population models in one of the environmental science or biology classes they may take at a university, college, or high school. They may also encounter population models in mathematics classes. To help people understand and explore population models, many people and organizations have created computer programs that demonstrate population models. There are multitudes of these free programs available on the Internet.

At California State University, Channel Islands (CSUCI), some of the professors use such computer programs to help their students learn about population models. However, the programs they have found on the Internet are scattered and sometimes unstable. Links to websites may change without notice, causing frustration for the teachers and students.

This Master's Thesis Project involved the creation of an interactive population models website for CSUCI. The address of the website is http://esrm.csuci.edu/PopulationModels.htm . This website will be maintained by CSUCI, so that it will always be available for use by its students. The website currently includes nine different aspatial population models. Each model page contains a Java applet, which is an embedded program that runs through a web browser. One of these applets can be seen in Figure 1. Each webpage explains the model that is being used, starting with the basics and building an explanation of why and how the model works. This website can be used as an interactive learning tool. It was designed to be used by people with a minimal level of mathematical knowledge, but the webpage explanations contain sections that will be more easily understood by people with some knowledge of math.

This website can be used in several ways. Teachers in a classroom can display the website and use the program to demonstrate various models. Students can work independently to learn about the models and interact

FIGURE 1. Random Harvesting Model Applet (Example 1)



with them. Teachers can create their own assignments for students that the students complete by using the programs, or they can instruct the students to read the pages and perform the activities described on the website. The website can be used for its ecological and biological significance, or simply for the mathematical aspect of learning about differential equations.

While this website was designed specifically for the teachers and students of CSUCI, it is available for free to anyone in the world through the Internet. It can certainly be used by other universities and even high schools. Individuals not associated with academia may find it of interest as well.

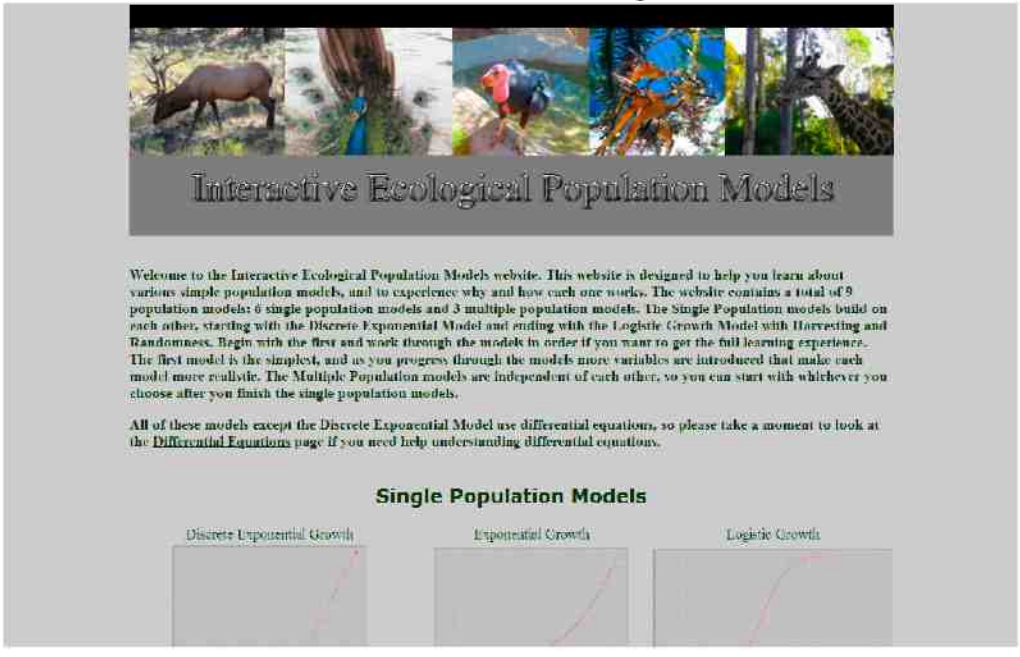
This paper contains the contents of the website in Section 2. Section 3 details some of the technical aspects of the website, and Section 4 explains the program and contains some examples of the program code. The equations that were used for the Balanced Ecosystems program are derived in Section 5. Section 6 explains the Runge-Kutta Method of Approximation for systems of differential equations, which was used in the program to obtain solutions for the models. Section 7 briefly discusses Stochastic Differential Equations (SDE's), because SDE's were used for one of the models. Section 8 contains some possible future extensions and ideas for the program and website. The Java code for one of the programs is in Appendix A and Appendix B.

2. Website Content

2.1. **Homepage.** Figure 2 contains a screenshot of the website's homepage. The homepage of the website greets visitors with the following text:

> Welcome to the Interactive Ecological Population Models website. This website is designed to help you learn about various simple population models, and to experience why and how each one works. The website contains a total of 9 population models: 6 single population models and 3 multiple population models. The Single Population models build on

FIGURE 2. Homepage



each other, starting with the Discrete Exponential Model and ending with the Logistic Growth Model with Harvesting and Randomness. Begin with the first and work through the models in order if you want to get the full learning experience. The first model is the simplest, and as you progress through the models more variables are introduced that make each model more realistic. The Multiple Population models are independent of each other, so you can start with whichever you choose after you finish the single population models. All of these models except the Discrete Exponential Model use dif-

ferential equations, so please take a moment to look at the Differential Equations page if you need help understanding differential equations.

The last line above contains a link to the Differential Equations page. Below the welcoming paragraph are the single population models, and below those are the multiple population models. Each model's name is shown above a screenshot of the applet to help the user select a model. Both the title and the image of each model link to that particular model's page.

To avoid copyright infringement, all the photographs on the website are original photos taken by the Perkins family, and all rights are given to CSUCI with the project.

2.2. **Differential Equations Page.** Differential equations are used for all of the models except the Discrete Exponential Model. It is important that the user have at least some understanding of what a differential equation is. The homepage encourages users to visit the Differential Equations page. The page is also mentioned in and linked to from several of the model pages. This page explains what a derivative is, and then gives the definition of a differential equation. The text is below.

We can use differential equations to describe the rate of change of a population's size at any time. In other words, the differential equation tells us how quickly or slowly the population is growing or decreasing.

Let us look at a fictional wolf population. At Year 1 there are 200 wolves and at Year 3 there are 544 wolves. The change in the number of wolves is $544 - 200 = 344$. However, the RATE of change can be found by dividing the change in the wolf population size by the change in time.

$$rate\ of\ change - \frac{change\ in\ size}{change\ in\ time}$$

$$- \frac{544 - 200}{3 - 1}$$

$$= \frac{344}{2} = 172$$

The rate of change we found was 172. On average, there are 172 wolves added every year.

The rate of change does not usually stay the same, however. At Year 2 there are 330 wolves and at Year 4 there are 896 wolves. We could find the rate of change for Year 4 by comparing it to one of the previous years.

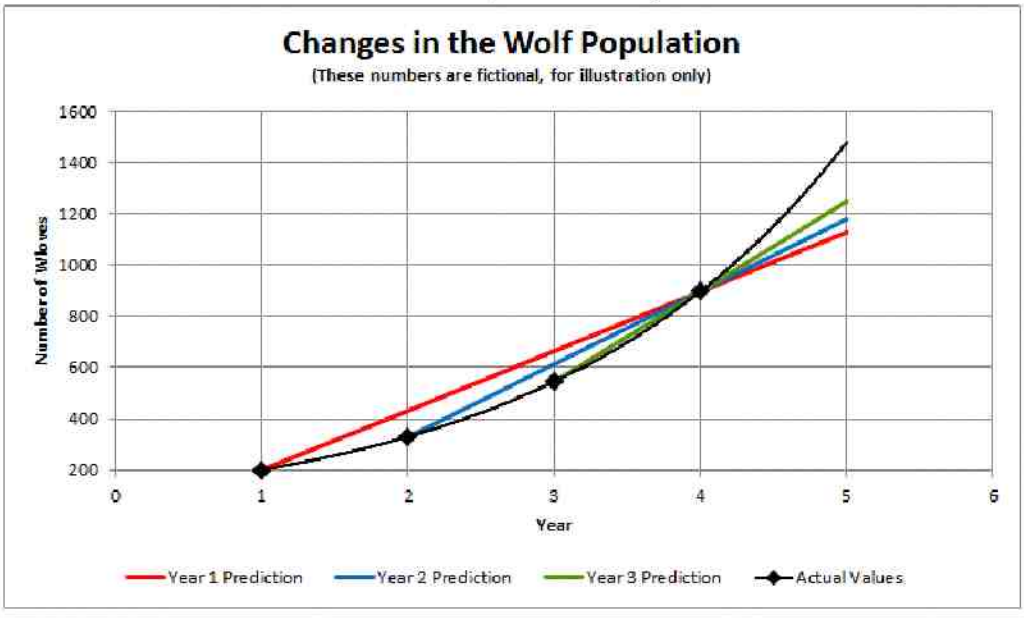
$$rate\ of\ change\ between\ Year\ 1\ and\ Year\ 4 = \frac{896 - 200}{4 - 1} = \frac{696}{3} = 232$$

$$rate\ of\ change\ between\ Year\ 2\ and\ Year\ 4 = \frac{896 - 330}{4 - 2} = \frac{566}{2} = 283$$

$$\textit{rate of change between Year 3 and Year 4} \quad \frac{896-544}{4-3} \quad \frac{352}{1} \quad 352$$

Using different years, we get different rates. Look at the graph below [Figure 3]. The black dots are the values we were given for the number of wolves. The colored lines show how the graph would look if we used the rates of change we calculated above. We are looking for the rate of change at Year 4. As we move the starting point closer to Year 4, the colored lines get closer to the actual graph at Year 4.

FIGURE 3. Wolf Graph in Differential Equations Webpage



If we continued moving the starting point closer to Year 4, we would eventually get the INSTANTANEOUS rate of change. Let us put this into mathematical terms. Let Δt be the change in time, and let Δp be the

change in the population size of the wolves over that time period. Then

$$rate\ of\ change - \frac{\Delta p}{\Delta t}$$

To get the instantaneous rate of change, we want to make Δt very small, infinitely small. To do this, we say that we take the limit as Δt approaches 0.

$$\lim_{\Delta t \to 0} \frac{\Delta p}{\Delta t} - \frac{dp}{dt}$$

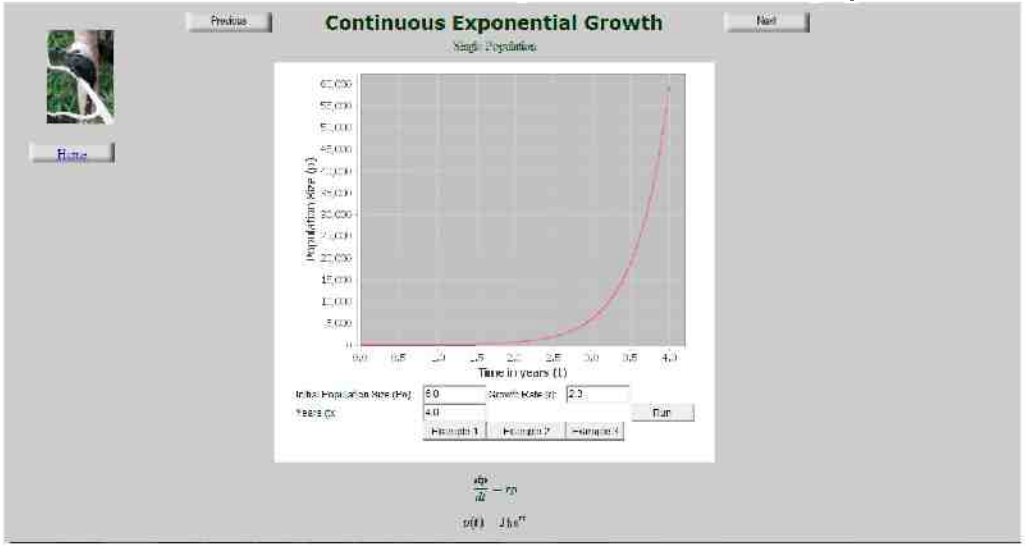
Here $\frac{dp}{dt}$ means the instantaneous rate of change in the population size at time t, and is called the "derivative" of p. If we have an equation that has both p and a derivative of p, it is called a "differential equation". If we let $p =$ the population size at time t, we could have the following differential equation:

$$\frac{dp}{dt} - 0.5p$$

The rate of change at any time is based on the population size at that time. Since the population size is not staying the same, neither is the rate of change. The rate of change at Year 1 is 0.5(200)=100, the rate of change at Year 2 is 0.5(330)=165, and the rate of change at Year 4 is 0.5(896)=448.

Now you know what a differential equation is. On this website we use differential equations to describe the instantaneous rate of change of a population's size. Differential equations are used not only for population modeling, but also in other fields such as physics, chemistry, and economics.

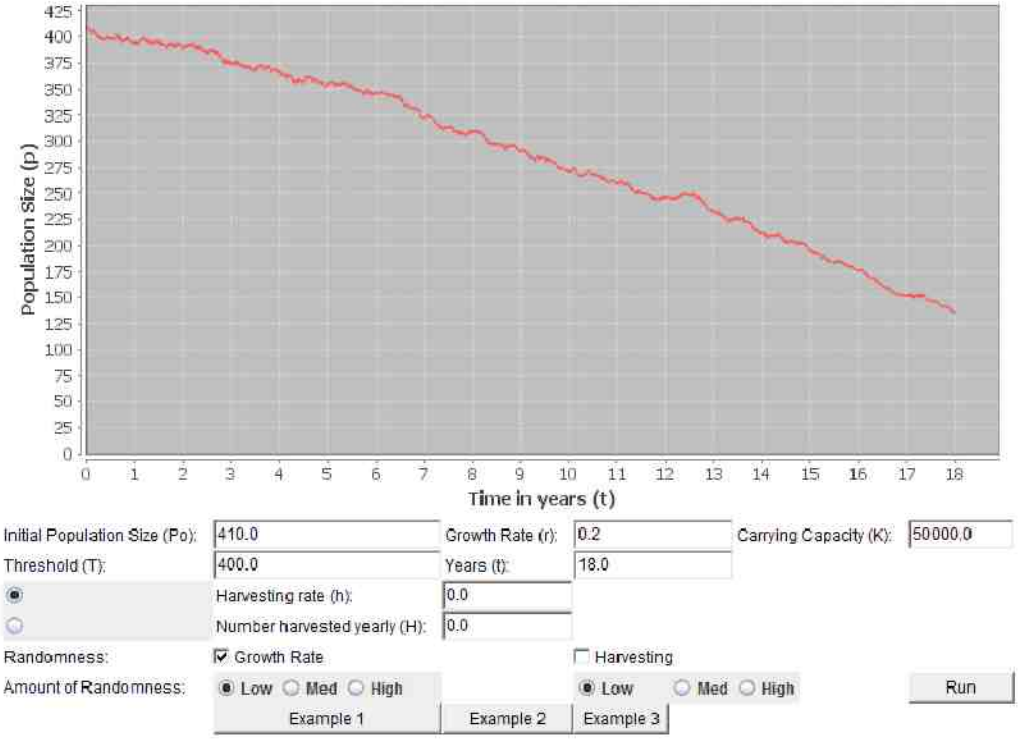
FIGURE 4. Continuous Exponential Model Webpage



2.3. **Model Webpage Overview.** Each model page follows a standard layout, and an example can be seen in Figure 4. At the top and bottom of the page are "Next", "Previous", and "Home" buttons to help the user navigate through the site. The applet is centered at the top of the page, preloaded with one of the examples. All of the applets contain a graph of the current model, text boxes for the user to change the parameters, a "Run" button, and three "Example" buttons. Some of the applets have additional

features such as radio buttons and check boxes. The most complex applet can be seen in Figure 5.

FIGURE 5. Logistic Growth with Harvesting and Randomness Applet



Below the applet is the equation or equations used for the model, along with a short description of all the variables used. For example, in the Logistic Growth Model with a Threshold, it has the following:

$$\frac{dp}{dl} = -rp\left(1 - \frac{p}{K}\right)\left(1 - \frac{p}{T}\right)$$

p − population size, r − growth rate, K − carrying capacity, T − threshold − minimum viable population

Below this is the main text of the webpage. We build each model logically, and explain the pieces from a mathematical perspective and an ecological perspective. Here is an excerpt from the Lotka-Volterra Model that demonstrates this.

> Next we have to find the predator equation. We start by making an equation that does not involve the prey. What does the predator population rate of change look like if there are no prey? They would simply die out. If we let d be the death rate of the predators in the absence of prey and restrict d to a positive number, the equation we get is
>
> $$\frac{dy}{dt} - -dy$$
>
> When there are prey available, they will have a positive effect on the rate of change of the predator population. The Lotka-Volterra model assumes that this effect will be proportional to the number of predators and also the number

of prey. When there are more prey, the predator population will increase. When there are more predators, the predator population will increase more because they produce more offspring. Let c be the constant that relates the increase in predators with the number of predators and prey. Then the change will be cxy. Our equation becomes

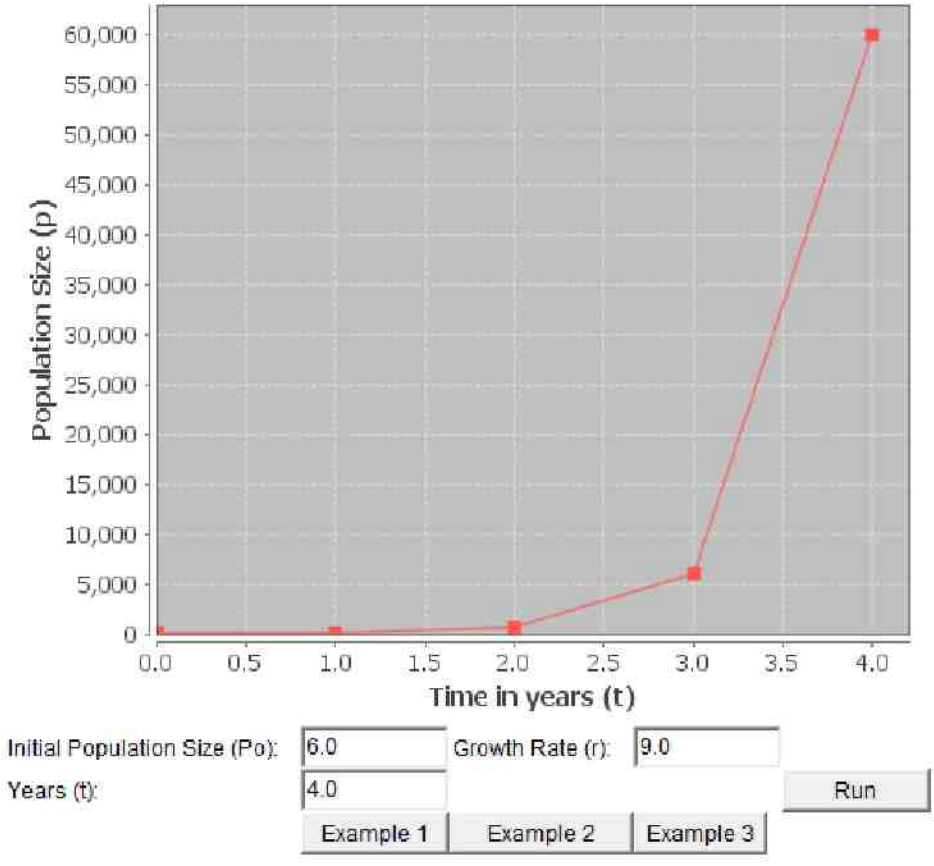
$$\frac{dy}{dt} = -dy + cxy$$

There are examples embedded in the text that correspond with examples in the applet, and the user is instructed in the text to click the appropriate example button. All pages encourage the user to experiment with the applet, and some give specific instructions on what to try. In the Harvesting Model, for example, the text says "Try adjusting the harvesting amount or the harvesting rate to find a sustainable harvest."

The text of all the webpages is contained in the following sections. Whenever the user is instructed to click an example button, a reference to a figure is included so the reader may see screenshots of the examples. Any links in the text that connect the pages have been reproduced here by adding a note with the proper section number.

2.4. **Model Webpages.**

FIGURE 6. Discrete Exponential Model Applet (Example 1)



2.4.1. *Discrete Exponential Model.*

$$p_{t+1} \quad (1+r)p_t$$

p_t = the population size at time t, p_{t+1} = the population size at time t + 1,

r growth rate

All population models are simplified representations of the natural world. We will start our population modeling with single population models. Single population models consider a population to be isolated from all other populations, which is very rarely true to reality. The simplest model is the Discrete Exponential Model.

It is impractical to try to create a model that accounts for everything that interacts with a population. Instead, each mathematical model uses select factors considered most influential, and assumes that only those factors are present. All the single population models we will see here share three factors in particular, and these are the three factors that make up the exponential model: birth rate, death rate, and current population size.

Suppose there are 60 rabbits today and 600 rabbits a year from now. Why did the number change? The most obvious answer to this question is that many rabbits were born. Maybe a less obvious cause of change was rabbits dying. Clearly, how many rabbits will be alive in a year depends on both how many are born and how many die.

Assume that for each rabbit now, 10 rabbits will be born this year. Then the "birth rate," the rate at which rabbits are born, is 10. If there are 5 rabbits now, there will be 50 rabbits born within a year. If there are 50 rabbits now, there will be 500 rabbits born within a year. Notice that how many rabbits are born depends on how many rabbits there are to begin

with. To find the number of rabbits born during a given year, we multiply b (the birth rate) by p_t (the number of rabbits alive at time t, the beginning of the year) to get bp_t. Similarly, the number of rabbits that die each year depends on how many were alive at the beginning of the year. To find the number of rabbits that die during the year we multiply d (the "death rate") by p_t to get dp_t.

We can put this information together to get an equation that tells us how many rabbits will be alive in a year. Let p_t be the population size at time t, p_{t+1} be the population size at time t+1 (1 year from time t), b be the birth rate, and d be the death rate. The number of rabbits alive at time t+1 equals the number of rabbits alive at time t plus the number of rabbits born minus the number of rabbits that died.

$$p_{t+1} = p_t + bp_t - dp_t$$

We can factor out p_t to get

$$p_{t+1} = (1 + b - d)p_t$$

We can group together the birth rate and the death rate to get the "growth rate," r.

$$r = b - d$$

Then

$$p_{t+1} \quad (1+r)p_t$$

This equation tells us how many rabbits there are each year. For example, suppose right now at time $t \quad 0$ there are 6 rabbits (so $p \quad 6$) and their growth rate per year is 9. We can find out how many rabbits there will be every year by plugging these numbers into the equation above.

$$p \quad (1+9)(6 \quad 60$$

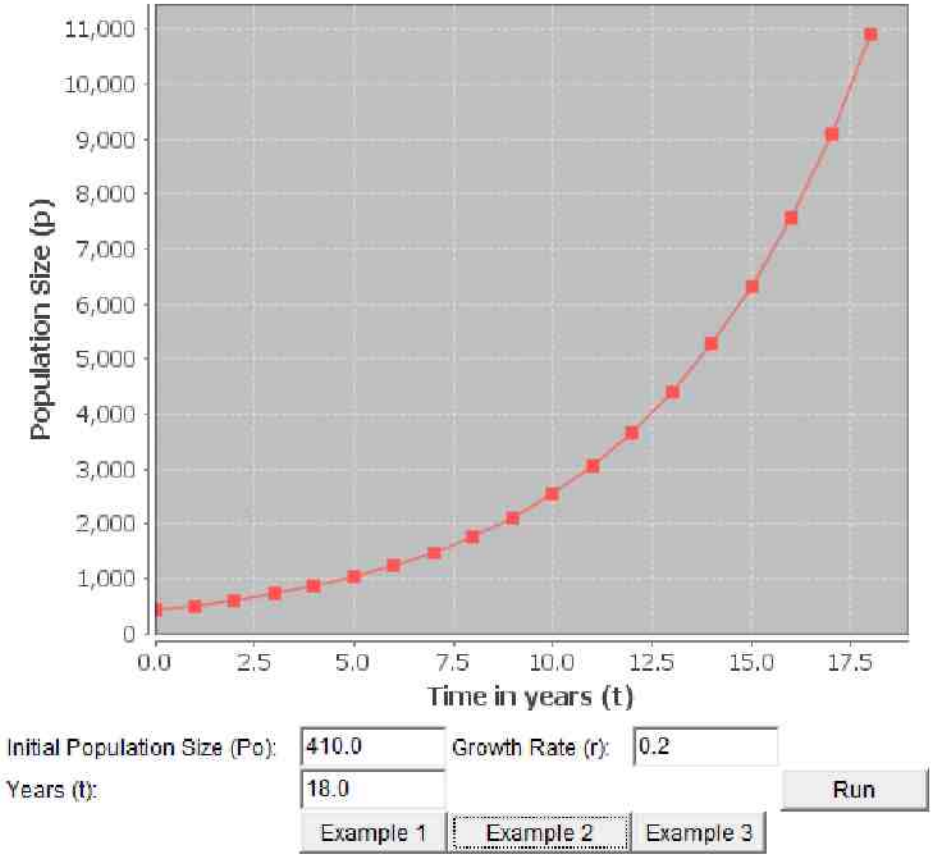
$$p_2 \quad (1+9)(60 \quad 600$$

$$p_3 = (1 \mid 9)(600 = 6000$$

$$p_4 \quad (1+9)(6000 \quad 60000$$

We find that after 4 years, there will be 60,000 rabbits. Click Example 1 [Figure 6] above to see the graph of the rabbit population size changing over time. If you click on the chart, you can hover over the points to see their values. The rabbits had a very high growth rate, so their numbers rose very quickly. Other populations grow much slower. Click Example 2 [Figure 7] to see a slower growing population, graphed over 18 years. Note that the population axis changes scale automatically. Example 3 [Figure 8] shows a population of bison growing over a period of 40 years. Notice

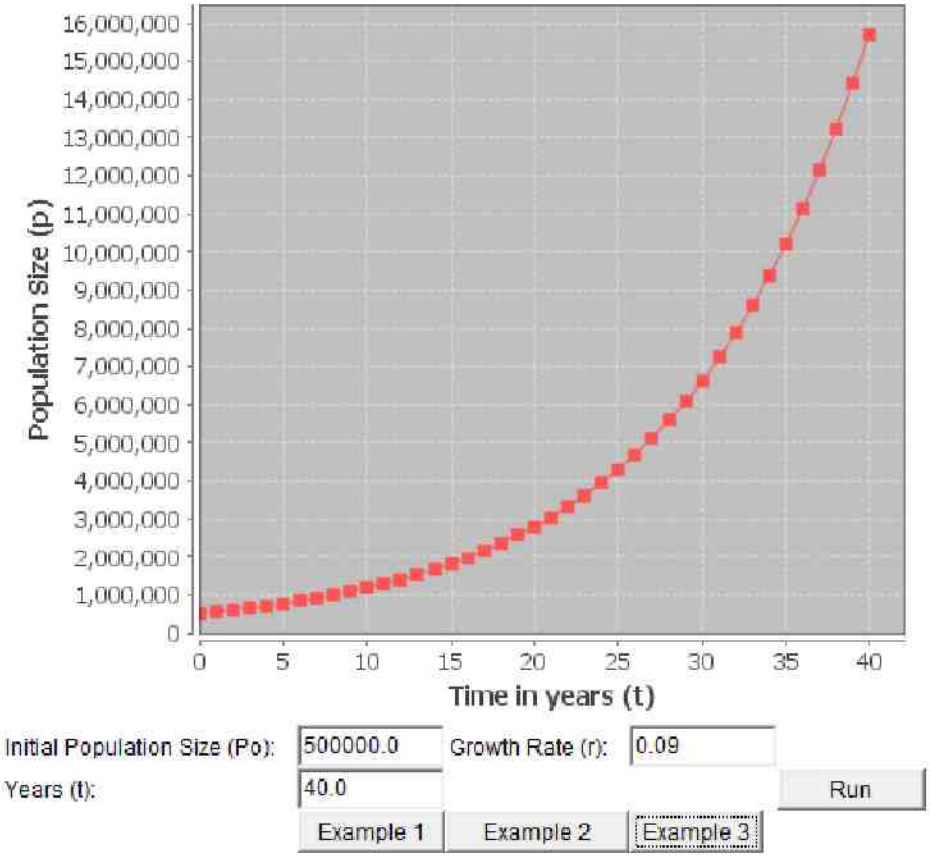
FIGURE 7. Discrete Exponential Model Applet (Example 2)



how the curves look similar in each case. Experiment with the program by changing the values and seeing what happens. (The numbers used for the animal examples are extremely rough estimates, used only for illustrating the model.)

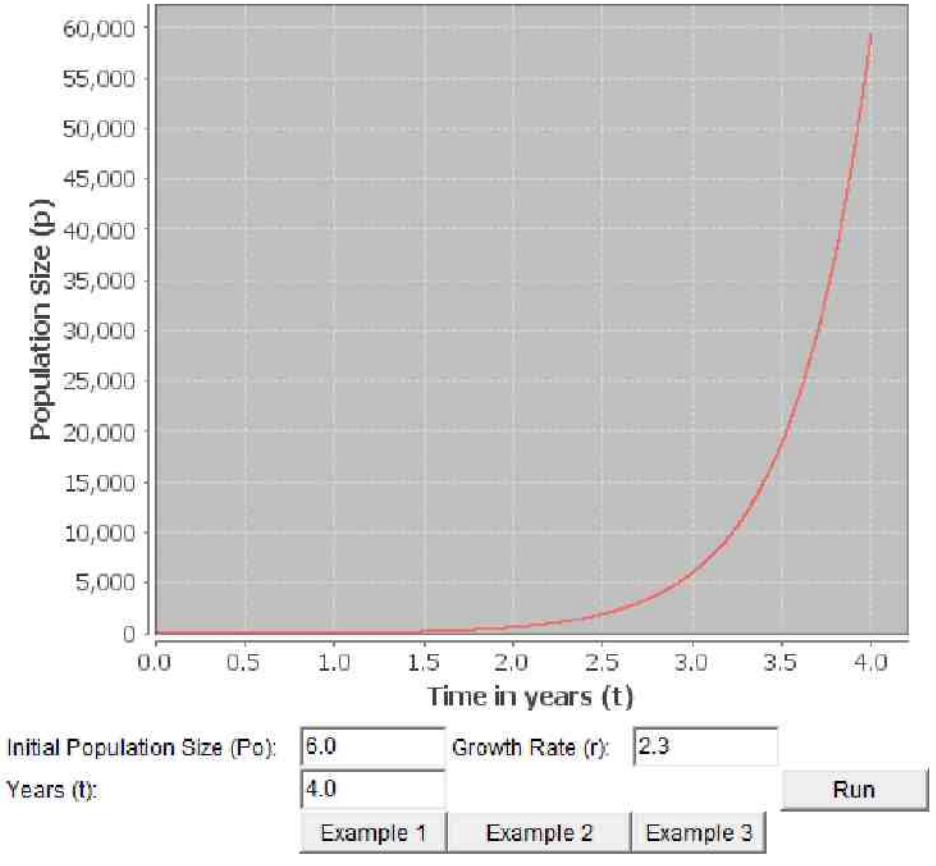
This model is a discrete model. This means it supposes that all the births and deaths occur at the same time, right at the beginning of the year. This causes a problem, because only the rabbits that are alive at the beginning

FIGURE 8. Discrete Exponential Model Applet (Example 3)



of the year can give birth or die during that year. In nature, a rabbit could be born at the beginning of the year and start reproducing partway through the year, but that is not possible in this model. The rabbit has to wait until the beginning of the next year to have offspring. To change this restriction, we typically use continuous models instead of discrete models, especially with larger population sizes. To see the Continuous Exponential Model, click Next.

FIGURE 9. Continuous Exponential Model Applet (Example 1)



2.4.2. *Continuous Exponential Model.*

$$\frac{dp}{dt} = rp$$

$$p(t) \quad P_0 e^{rt}$$

$p(t$ = the population size at time t, t = time, r = growth rate, $P_$ = the starting population size

The Exponential Model assumes that a population is in isolation. The only factors that affect the population size are the birth and death rates and the current population size. This model is a basis for all the other models we will explore.

While the previous page addressed the Discrete Exponential Model [Section 2.4.1], this page is about the Continuous Exponential Model. In a discrete model, the changes happen only once every year, or once every time period. This distinguishes it from a continuous model, which allows for animals to be born or die at any time.

We saw in the Discrete Exponential Model that the change in population size is affected by the birth rate, death rate, and current population size. We combine the birth rate, b, and the death rate, d, to get the growth rate, r.

$$r \quad b - d$$

The equation we found for the Discrete Exponential Model is below. Here p is the population size at time t, p_{t+1} is the population size at time t + 1 (1 year from time t), and r is the growth rate.

$$p_{t+1} = (1 + r)p_t$$

OR

$$p_{t+1} \quad p_t + r \quad p_t$$

This equation tells us that the population size in one year equals the population size now, plus r times the population size now. Notice that the change in the population size is just $r \quad p_t$. We could have written an equation like this instead:

$$\textit{Change in one year} \quad r \quad p_t$$

In the discrete model it made more sense to write the equation the way we originally showed it, but in a continuous model it will be easier to write an equation for the rate of change of the population size. The discrete equation allowed us to step one year at a time, and calculate the change at each step. In the continuous model, we are no longer jumping over the whole year but moving continuously through it. For this reason, we will use a differential equation for the model. (If you do not know what a differential equation is, click here [Section 2.2].)

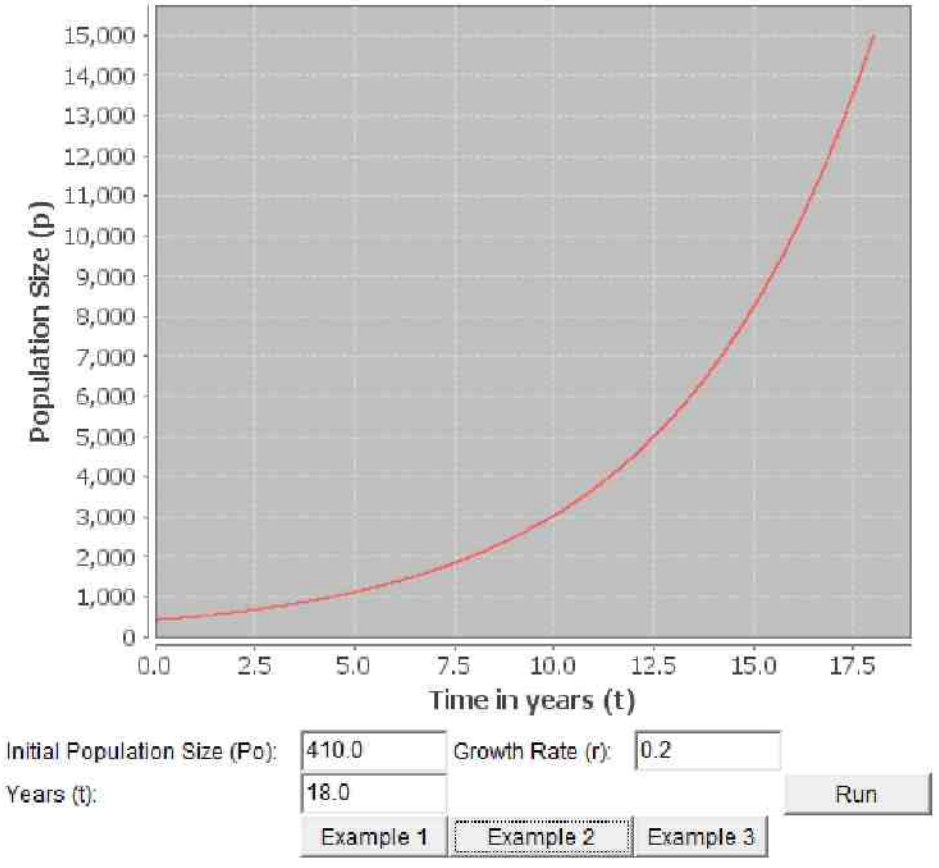
Let p be the population size at time t, t be the time (in years, in this case), and r be the growth rate. Then the rate of change of the population size (written as $\frac{dp}{dt}$) is:

$$\frac{dp}{dt} \quad rp$$

If we have an initial population P_0 (the size of the population at time 0, the beginning), we can solve this equation to get $p(t) = P_0 e^{rt}$. (Note that this is a function "p of t," not "p times t".) Now we have a function that relates time and population size. We can replace the variables with numbers to find the population size at any time. For example, if Population A has a growth rate of 0.2 and an initial population of 410, at time 2 we will have $p(2) = 410e^{(0.2)(2)} \approx 612$. We can also find the population size at time 2.5, halfway between 2 and 3. This would be $p(2.5) = 410e^{(0.2)(2.5)} \approx 676$. Click Example 2 [Figure 10] to see a graph of this.

There are a few more differences between discrete models and continuous models that we need to note. Let us revisit the rabbit example from the Discrete Exponential Model. If we have an initial population of 6 rabbits with a growth rate of 9, then after 1 year this model tells us there will be $p(1) = 6e^{(9)(1)} \approx 48{,}618$ rabbits. We started with the assumption that for each rabbit now, 10 rabbits will be born in a year. So how could 6 rabbits turn into 48,618 in one year? For one thing, the continuous model allows for fractions of an animal. If 1 rabbit can have 10 offspring, one-half of a rabbit can have 5 offspring. This does not make physical sense. The continuous model also assumes that as soon as an animal is born, it begins to reproduce. We know this is not how it actually works. Rabbits are

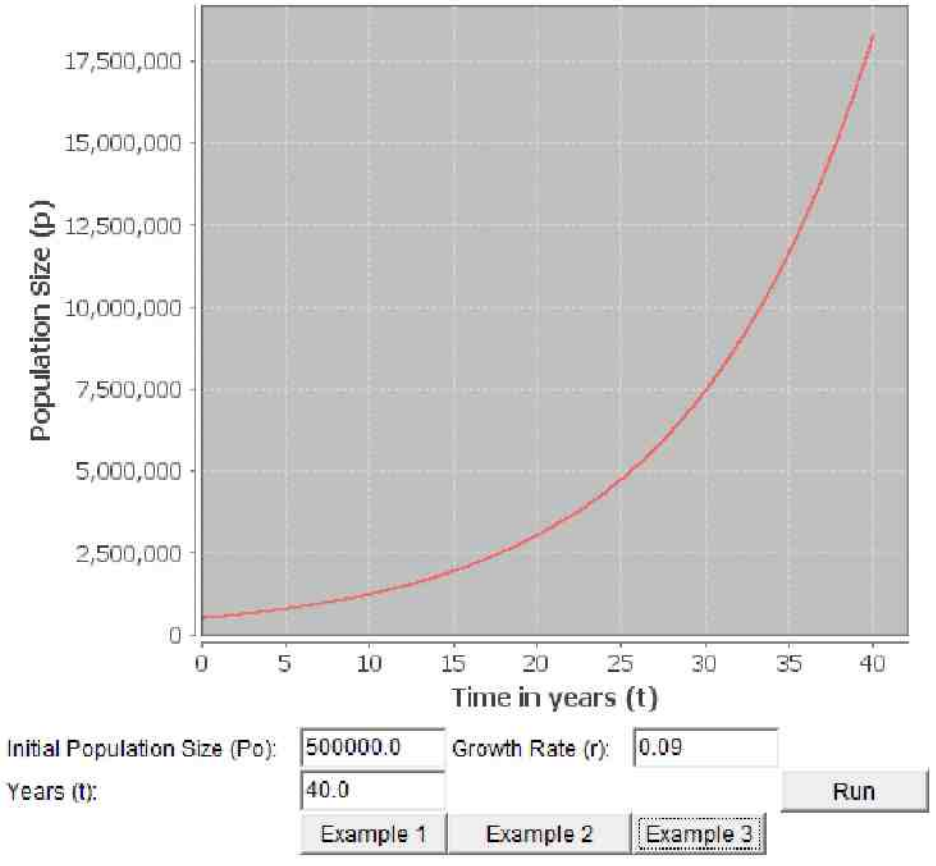
FIGURE 10. Continuous Exponential Model Applet (Example 2)



usually 6 or 7 months old before they begin to reproduce[28]. To adjust for these things, we have to change our growth rate.

We can say instead that the growth rate of the rabbits is 2.3. (This is the rate we find by letting $P_0 = 6$, $p(1) = 60$, and $t = 1$.) Then after 1 year, there will be $p(1) = 6e^{(2.3)(1)} \approx 60$ rabbits. In 4 years, there will be $p(1) = 6e^{(2.3)(4)} \approx 59{,}382$ rabbits. These numbers are similar to the numbers in the Discrete Model. You can compare the two models by clicking Example

FIGURE 11. Continuous Exponential Model Applet (Example 3)



1 [Figure 9] in this model and in the Discrete Exponential Model [Section 2.4.1]. Notice that the curve in this model is smooth. The population size is changing all the time, instead of just at the year marks.

The program above generates the graph using the equation $p(t) = P_0 e^{rt}$. You can experiment with the program to see what happens if you change the initial population or the growth rate. Predict what will happen when you:

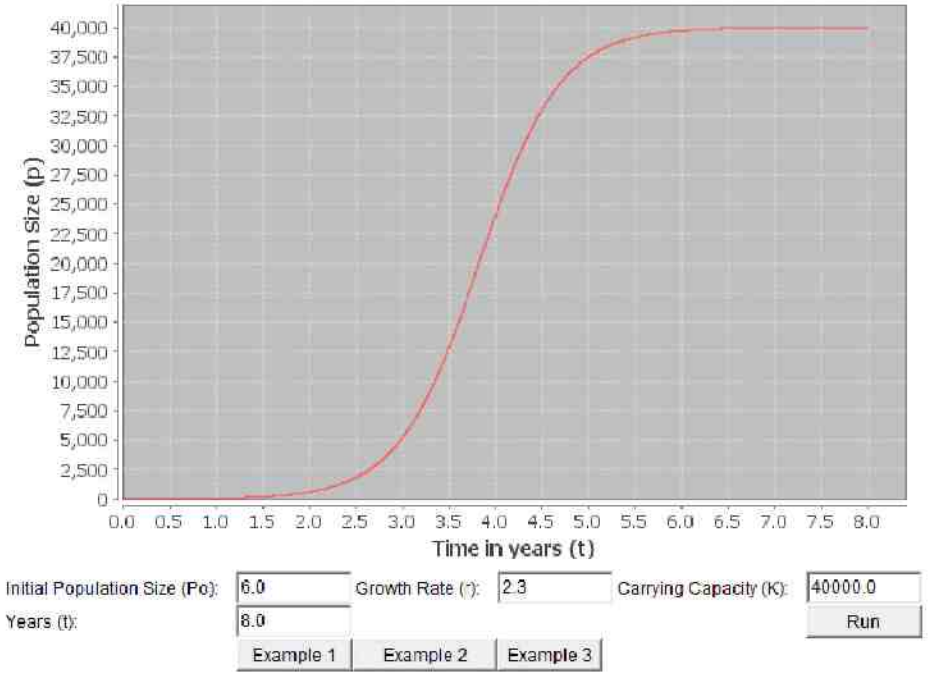
1. Increase the initial population.
2. Decrease the initial population.
3. Change the growth rate to 0.5.
4. Change the growth rate to -0.5.
5. Change the growth rate to 0.

After you have made your predictions, experiment with the program above to see what really happens. You can also click on one of the Example buttons to reset the program.

You should have observed that if the growth rate is 0, the population size never changes. If the growth rate is negative, more animals are dying than are being born, so the population eventually dies out.

As mentioned before, our exponential model assumes that there are no outside influences. This model is very simplified. Bacteria in a laboratory, for example, can be modeled fairly accurately using an exponential model, at least for a limited time. The problem with a model using exponential growth is that the population keeps growing forever. In actuality, something will make the growth stop, usually limited resources such as food. To learn about this, move on to the Logistic Growth Model [Section 2.4.3].

FIGURE 12. Logistic Model Applet (Example 1)



2.4.3. *Logistic Growth Model.*

$$\frac{dp}{dt} \quad rp\left(1-\frac{p}{K}\right)$$

p population size, r growth rate, K carrying capacity

Imagine a mad scientist creates a species of giant killer cockroaches that cannot be killed. They breed like crazy, doubling in numbers every day. They spread over the planet, eating everything and everyone. Eventually they will have covered the entire world and eaten everything. And then they will die, because they have used up all the resources of the entire planet and they need more food to survive. No population can keep increasing forever.

One of the limitations we mentioned in our discussion of the exponential model is that the population continued to grow forever, which is not realistic. The logistic model accounts for limited resources, which prevent the population from growing indefinitely. Limited resources can include such things as food, nesting space, shelter, and water. An environment can only sustain a certain population size long-term. The size of the population that can be sustained is called the "carrying capacity." If a population exceeds this limit, they will start dying off. As they approach this limit, their growth will slow.

Let p be the population size, r the growth rate, and K the carrying capacity. The Logistic Growth Model says the rate of change in population size ($\frac{dp}{dt}$) at time t is:

$$\frac{dp}{dt} - rp\left(1 - \frac{p}{K}\right)$$

This equation starts out similar to the model for exponential growth, $\frac{dp}{dt} - rp$. Then it has another factor, $1 - \frac{p}{K}$. This piece might not make sense at first glance, but below we investigate how it works. Take a minute and try to answer the following 3 questions.

Example question: What does it mean if p is smaller than K? What do you expect will happen to the population size?

Example answer: This means the population size is smaller than the carrying capacity, so we expect the population to grow until it reaches the carrying capacity.

1. What does it mean if K is extremely large, close to infinity? What effect would you expect this to have on the population size?

2. What does it mean if p is larger than K? What do you expect will happen to the population size?

3. What does it mean if p is close to K? What effect would you expect this to have on the population size?

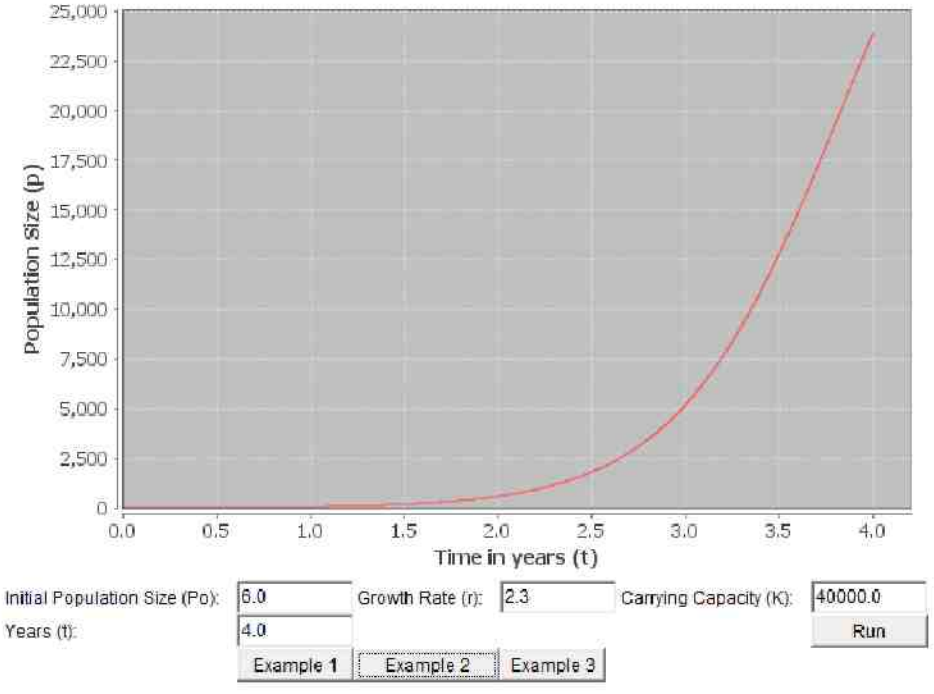
We will look at the answers to these questions and see that the model does exactly what we expect it to do.

1. If K is extremely large, the carrying capacity is high. The environment seemingly has no limit on how many animals it can sustain. We expect to see the population increase rapidly, exponentially in fact. This scenario implies taking the limit of the equation as K goes to infinity. (Do not worry if you are unfamiliar with limits. The paragraph below the limit equation will explain it.)

$$\lim_{K\to\infty} rp\left(1-\frac{p}{K}\right) = rp(1-0) = rp$$

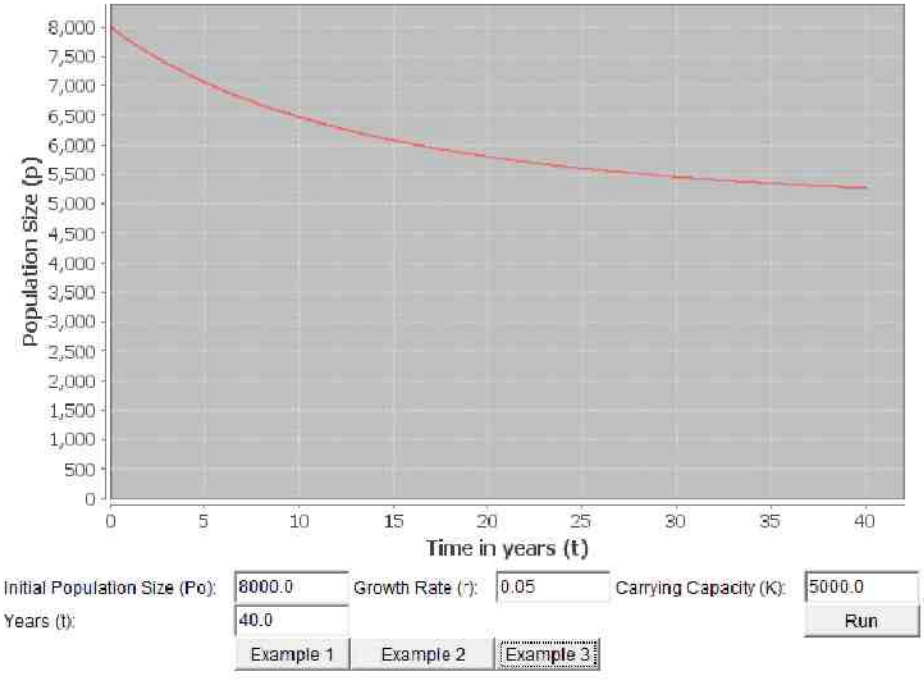
If K is very large compared with p, $\frac{p}{K}$ will be close to 0 (think about $\frac{100}{100{,}000{,}000}$=0.000001), and the equation will be close to $\frac{dp}{dt} = rp$. Perhaps

FIGURE 13. Logistic Model Applet (Example 2)



you noticed that this is just the exponential model. Recall that the exponential model does not account for limited resources. If K is very large, this simulates unlimited resources. So when p is small compared with K, we get exponential growth behavior. Click on Example 1 [Figure 12] above and look at the graph between Years 0 and 4. This part is very similar to exponential growth. Click on Example 2 [Figure 13] to zoom in to just this time range. You can compare it to actual exponential growth by going back to the Continuous Exponential Growth Model [Section 2.4.2] and clicking Example 1 [Figure 9].

FIGURE 14. Logistic Model Applet (Example 3)



2. If p is larger than K, this means the population size is larger than the carrying capacity. There are too many animals for the environment to sustain. We expect that the population size will decrease. In the model, when p is larger than K, $\frac{p}{K}$ will be greater than 1, so $1-\frac{p}{K}$ will be negative. This will cause a NEGATIVE change. The population size will start declining, as we would expect. You can see an example of this by clicking Example 3 [Figure 14].

3. If p is close to K, this means the population size is close to the carrying capacity. The environment is reaching its limit of how many animals it can sustain. We expect the population size to stabilize, to stop increasing or

decreasing. Looking at the model mathematically, this scenario gives:

$$\lim_{p \to K} rp\left(1 - \frac{p}{K}\right) = rp(1 - \frac{K}{K}) = rp(1-1) = rp(0) = 0$$

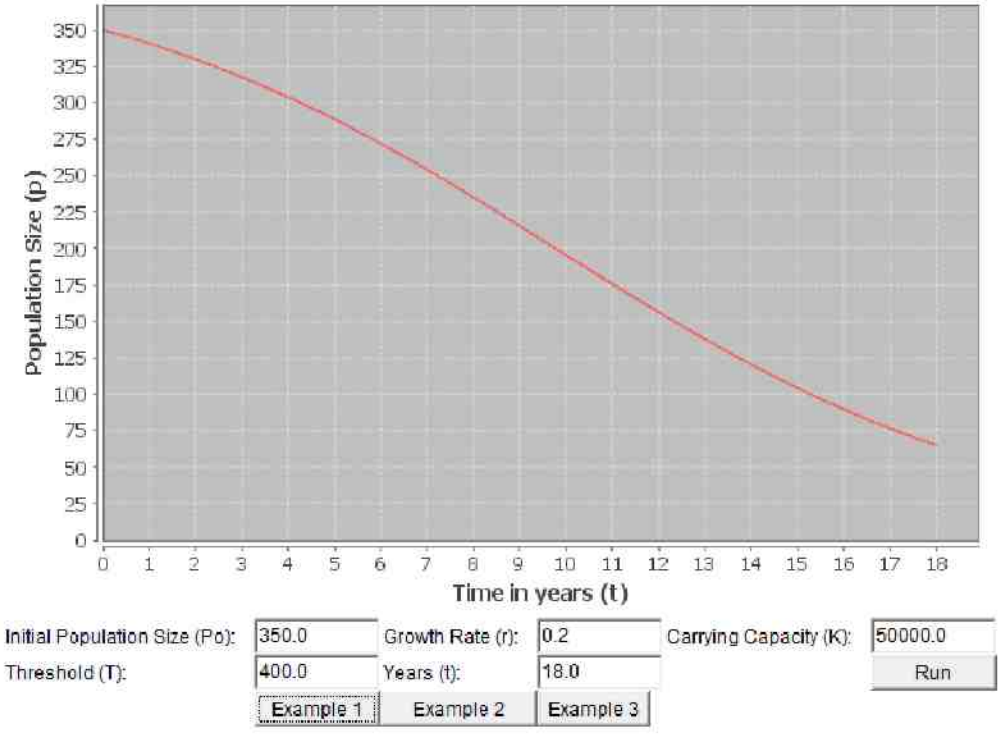
If p grows until it is very close to K, $\frac{p}{K}$ will be approximately 1. Then $1 - \frac{p}{K}$ will be $1 - 1 = 0$. The change will become 0. The population will stop growing and stay where it is. This means the population has reached its sustainable point, its carrying capacity. Click on Example 1 again and look at the last part of the graph, where the population size is approaching the carrying capacity.

Experiment with the program above by changing all the variables. If you need to reset it at any time, click one of the Example buttons.

The differential equation used in the Exponential Model was easily solved, but most differential equations are difficult or actually impossible to solve. This program, and all the ones to follow, use the Runge-Kutta Method to find approximate solutions.

When you are done with this page, click Next to move on to the Logistic Growth Model with a Threshold.

FIGURE 15. Threshold Model Applet (Example 1)



2.4.4. *Logistic Growth Model with a Threshold.*

$$\frac{dp}{dt} \quad -rp\left(1-\frac{p}{K}\right)\left(1-\frac{p}{T}\right)$$

$$\frac{dp}{dt} = rp\left(1-\frac{p}{K}\right)\left(\frac{p}{T}-1\right)$$

p population size, r growth rate, K carrying capacity,

T threshold minimum viable population

We have seen now that the population size can decrease if it gets too large. It can also decrease if it gets too small. If there are not enough animals,

they can be susceptible to extinction by natural disasters or diseases. There is also the possibility that if the population is too small, it will be unlikely that they will reproduce. Suppose we have a population of 100 wolves, spread out over the whole state of Texas. The population might die out because two wolves of the opposite sex never meet. Or if they do meet, they do not happen to mate and conceive on that encounter. Consider also this scenario: The last living female elephant gives birth to two male elephants, and then dies. That population is not going to grow anymore.

These situations involve a lower limit on the population, below which the population will start dying out. This limit is called the "threshold," or the "minimum viable population."

To create the model with a threshold, we start with the Logistic Growth Model [Section 2.4.3].

$$\frac{dp}{dt} \quad rp\left(1 - \frac{p}{K}\right)$$

$p =$ population size, $r =$ growth rate, $K =$ carrying capacity

Then we multiply on a new factor, $(\frac{p}{T} - 1)$, where T is the threshold. We get

(1) $$\frac{dp}{dt} \quad rp\left(1 - \frac{p}{K}\right)\left(\frac{p}{T} - 1\right)$$

If we factor a negative out, this can be rewritten

$$\frac{dp}{dt} \quad -rp\left(1-\frac{p}{K}\right)\left(1-\frac{p}{T}\right)$$

The second equation is how it is usually written, so that the threshold factor looks like the carrying capacity factor. Notice the negative sign in front. The first equation is equivalent, and is presented here because it is easier to explain. Take a minute to think about the following questions.

Questions:

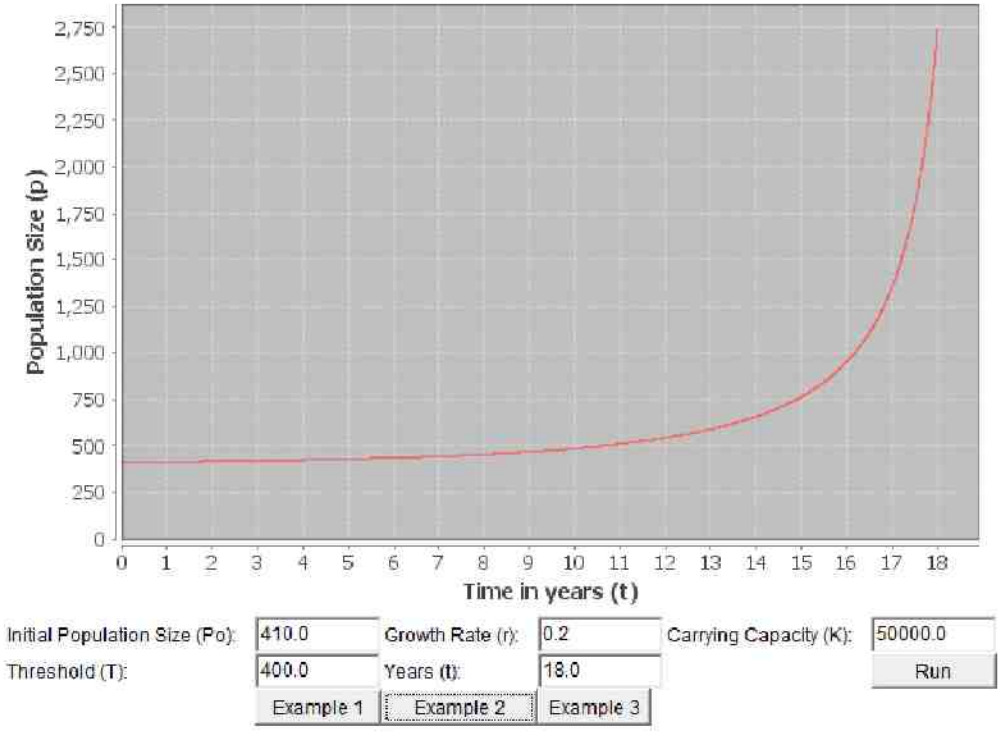
1. What does it mean if p is smaller than T? What will happen to the population size?

2. What does it mean if p is larger than T? What will happen to the population size?

Answers:

1. If p is smaller than T, this means the population size is below the threshold. In Equation 1, $\frac{p}{T}$ will be less than 1, and $\frac{p}{T} - 1$ will be negative. The threshold will always be less than the carrying capacity, so if the size is below the threshold it is also below the carrying capacity. As a result, $1 - \frac{p}{K}$ will be positive. Multiplying the two pieces together results in a negative, and the population size will start declining. This is the effect of

FIGURE 16. Threshold Model Applet (Example 2)

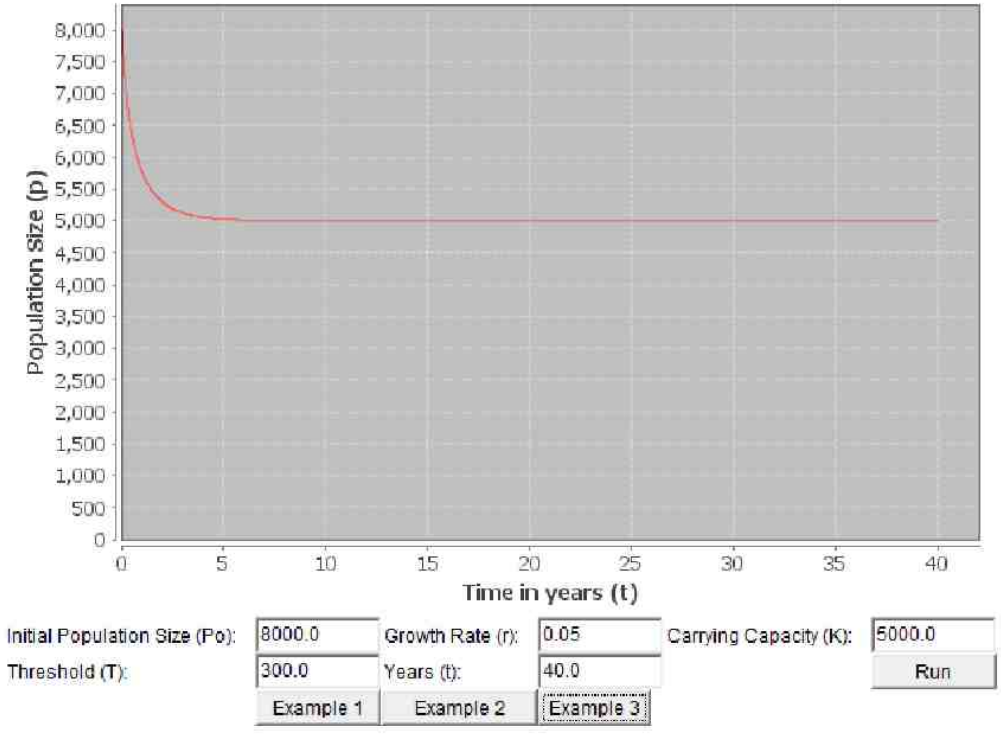


the threshold. If the population size falls too low, they will die out. Click on Example 1 [Figure 15] to see a population that is below its threshold.

2. If p is larger than T, then $\frac{p}{T}$ will be greater than 1, so $\frac{p}{T} - 1$ will be positive and the population size will increase, assuming we are not above carrying capacity. Example 2 [Figure 16] shows the same population as Example 1, except it starts out a little larger. Since it is above its threshold, it will survive.

This model still includes the carrying capacity, which is the maximum amount sustainable by the environment. If the population is larger than

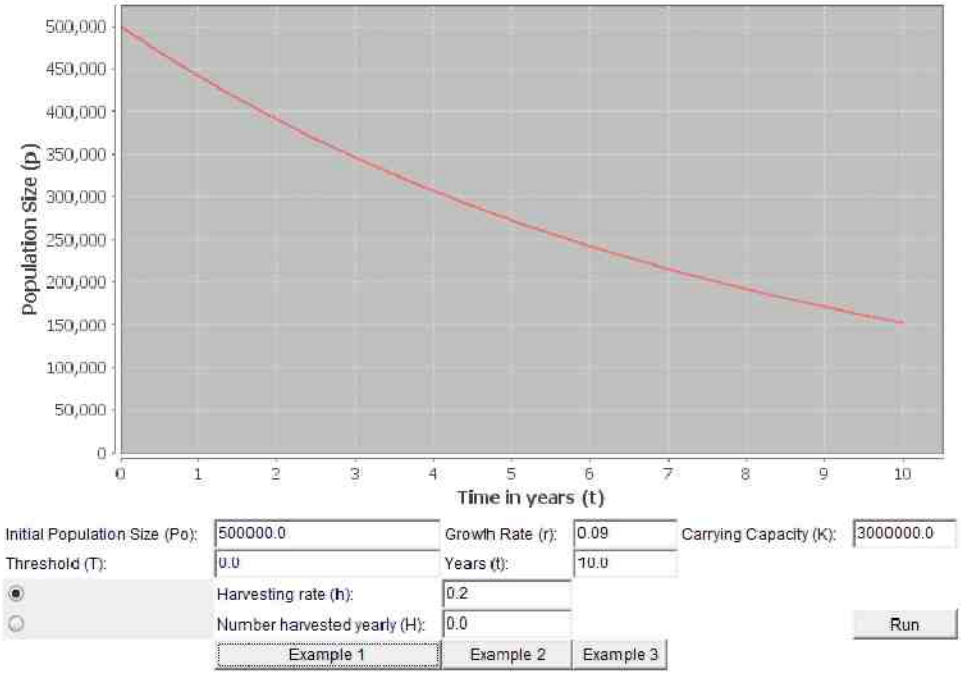
FIGURE 17. Threshold Model Applet (Example 3)



the carrying capacity, it will decrease, just as in the Logistic Growth Model [Section 2.4.3]. You can see an example of this by clicking Example 3 [Figure 17].

Experiment with the program above to see the effect of changes to the variables. Then you can move on to learn about a model with harvesting.

FIGURE 18. Harvesting Model Applet (Example 1)



2.4.5. *Logistic Growth Model with Harvesting.*

$$\frac{dp}{dt} \quad rp\left(1-\frac{p}{K}\right)-H \text{ (Constant Harvesting without Threshold)}$$

$$\frac{dp}{dt} \quad rp\left(1-\frac{p}{K}\right)-hp \text{ (Proportional Harvesting without Threshold)}$$

$$\frac{dp}{dt}=rp\left(1-\frac{p}{K}\right)\left(\frac{p}{T}-1\right)-H \text{ (Constant Harvesting with Threshold)}$$

$$\frac{dp}{dt} \quad rp\left(1-\frac{p}{K}\right)\left(\frac{p}{T}-1\right)-hp \text{ (Proportional Harvesting with Threshold)}$$

p population size, r growth rate, K carrying capacity,

T threshold, h harvesting rate, H amount harvested

The models we have seen until this point have assumed that a population is living in isolation. The death rate we have been using accounts for deaths that result from various natural causes, such as old age and accidents. In this model, we expand the causes of death by considering death by "harvesting." Also known as hunting, fishing, picking, and so forth, harvesting involves humans removing some members from the population. We need models with harvesting to see what effect we will have on a population. We will look at two possible ways to incorporate harvesting into a model.

The first model is Constant Harvesting. This model is used for removing a constant number of animals each year. For example, if we harvest 35,000 bison every year, regardless of how many bison there are, we would use constant harvesting. This model starts with either the Logistic Growth Model [Section 2.4.3] or the Logistic Growth Model with a Threshold [Section 2.4.4], and then the harvesting constant, H, is subtracted off.

Let p be the population size, r the growth rate, K the carrying capacity, T the threshold, and H the amount harvested.

Constant Harvesting

$$\frac{dp}{dt} = rp\left(1 - \frac{p}{K}\right) - H \text{ (Without Threshold)}$$

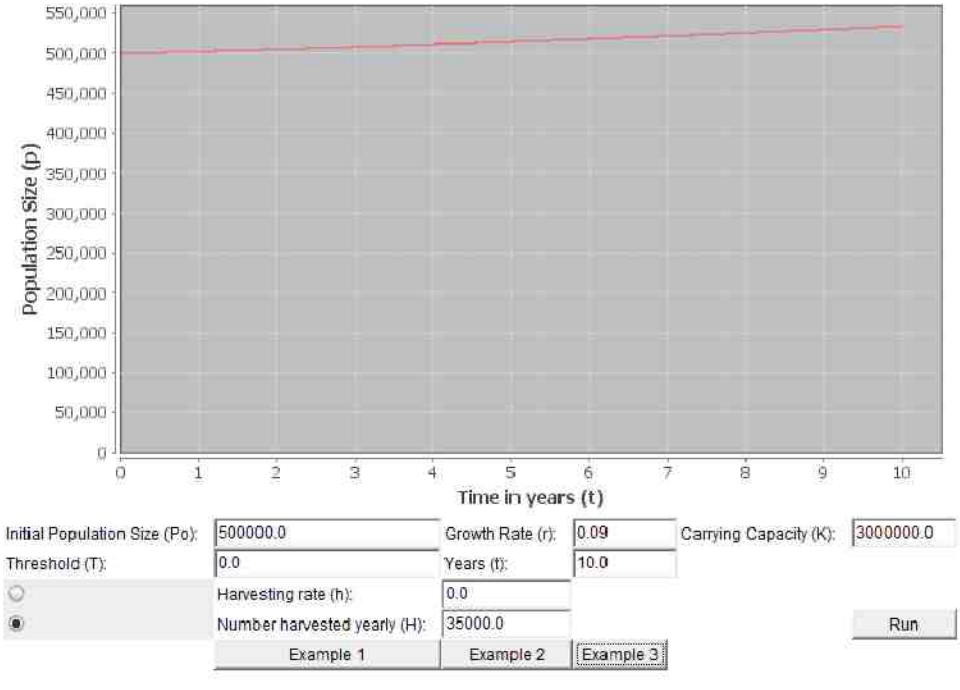
OR

$$\frac{dp}{dt} = rp\left(1 - \frac{p}{K}\right)\left(\frac{p}{T} - 1\right) - H \text{ (With Threshold)}$$

When you set the threshold to 0 in the program, this means there is no threshold, so the program uses the "Without Threshold" model. When the threshold is higher than 0, the program uses the "With Threshold" model.

Click Example 3 [Figure 19] to see the population of bison being harvested using constant harvesting. Change the number harvested yearly to 0 to see how the population would grow without the harvesting.

FIGURE 19. Harvesting Model Applet (Example 3)



The other model we will use is Proportional Harvesting. Instead of having a set amount to harvest every year, this model is used for harvesting a

PROPORTION of the population. When there are more animals we harvest more of them, and when there are less animals we harvest less of them. If we harvest 10% of the bison, the harvesting rate, h, would be 0.10. This model also starts with the Logistic Growth Model or the Logistic Growth Model with a Threshold, but we subtract a proportion by multiplying the current population size p, by the harvesting rate, h. So if there are 100,000 bison and we harvest 10% of them, that would be hp $(0.10)(100000)$ 10,000 bison harvested.

Let p be the population size, r the growth rate, K the carrying capacity, T the threshold, and h the harvesting rate.

Proportional Harvesting

$$\frac{dp}{dt} \quad rp\left(1-\frac{p}{K}\right) - hp \text{ (Without Threshold)}$$

OR

$$\frac{dp}{dt} = rp\left(1-\frac{p}{K}\right)\left(\frac{p}{T}-1\right) - hp \text{ (With Threshold)}$$

Click Example 1 [Figure 18] to see the population of bison being harvested using proportional harvesting. When we have a resource that we are harvesting, we want to make sure we do not harvest it into extinction. It can be difficult to find the right balance. If harvesting is higher than the

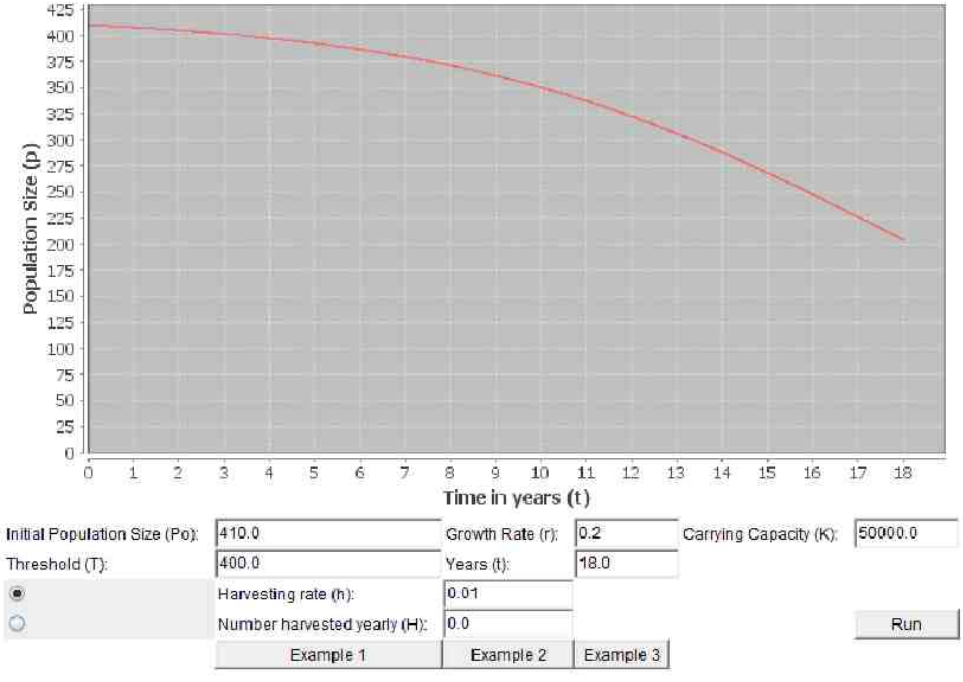
growth, we are overharvesting, and will drive the population to extinction. The bison population in Example 1 is being overharvested. We need to find a new harvesting rate. It seems logical to think that if the harvest rate equals the growth rate, this would be a sustainable harvest. However, the growth rate is the rate at which the population would grow without a carrying capacity or threshold. Try adjusting the harvesting amount or the harvesting rate to find a sustainable harvest.

When a population is near its threshold size, even slight harvesting can mean the difference between survival and extinction. Click Example 2 [Figure 20] above. This population started out above its threshold, but harvesting brought it below its threshold and the population will die off. Try increasing the initial population, and you will find that the population will survive if it starts with enough members.

When the harvesting balances with the growth, the population will stay at roughly the same size over time. The same amount can be harvested every year indefinitely. Click Example 3 [Figure 19] again to see the population of bison, which is being harvested at a sustainable rate.

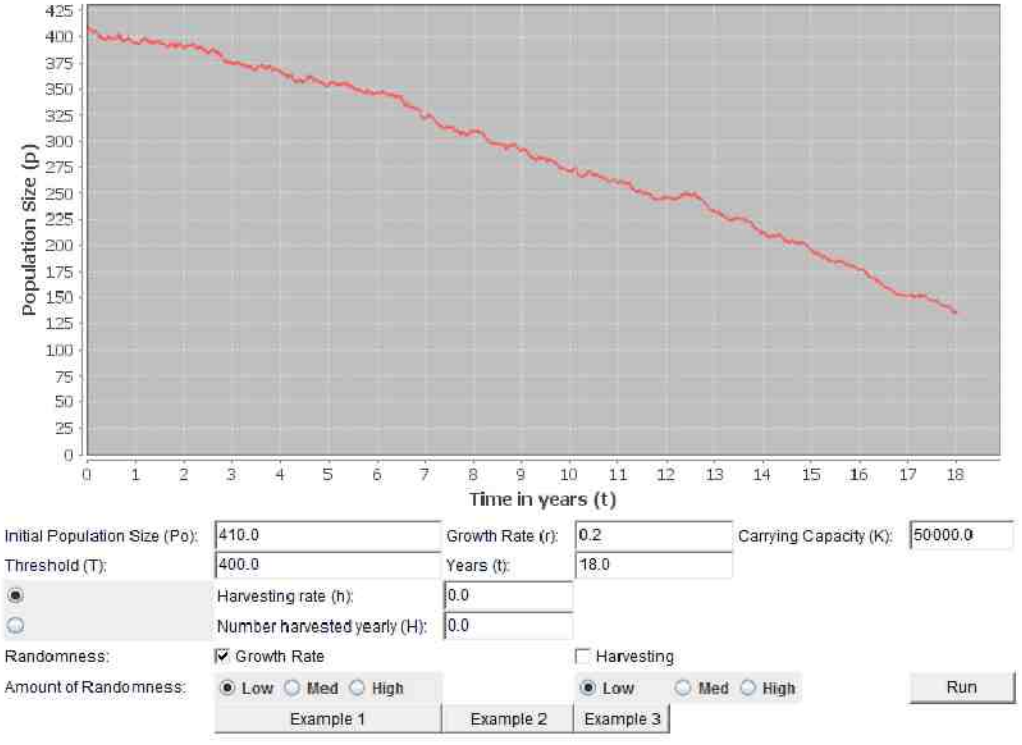
Of course, this model assumes that we can harvest exactly the number we want to. However, if we are using proportional harvesting, it can be difficult to know the exact number of animals available, so we may end up with a higher or lower proportion than we intended. Consider also a fisherman

FIGURE 20. Harvesting Model Applet (Example 2)



who sets out to catch a certain amount of fish, which has been deemed the correct sustainable amount. He may be unlucky and not catch as many as he wanted. Or he may catch too many. Real life is not as simple and predictable as this model, so the next model incorporates some randomness. Move on to the Logistic Model with Harvesting and Randomness [Section 2.4.6].

FIGURE 21. Randomness Model Applet (Example 1)



2.4.6. *Logistic Growth Model with Harvesting and Randomness.*

$$dp = \left[rp\left(1 - \frac{p}{K}\right)\left(\frac{p}{T} - 1\right) - H\right]dt \mid X_1 \mid Y_1$$

(Constant Harvesting with Threshold)

$$dp = \left[rp\left(1 - \frac{p}{K}\right)\left(\frac{p}{T} - 1\right) - hp\right]dt \mid X_1 \mid Y_2$$

(Proportional Harvesting with Threshold)

$$dp = \left[rp\left(1 - \frac{p}{K}\right) - H\right]dt \mid X_2 \mid Y_1$$

(Constant Harvesting without Threshold)

$$dp = \left[rp\left(1 - \frac{p}{K}\right) - hp \right] dt \mid X_2 \mid Y_2$$

(Proportional Harvesting without Threshold)

p = population size, r = growth rate, K = carrying capacity,

T = threshold, h = harvesting rate, H = amount harvested, X_1, X_2, Y_1, Y_2

are random variables

The world is unpredictable. We are never able to say with certainty exactly what will happen. We can say, for example, that a female rabbit has on average 10 offspring per litter. However, the number of offspring each rabbit actually produces in each litter is random. A rabbit may have 4 young, or 14.

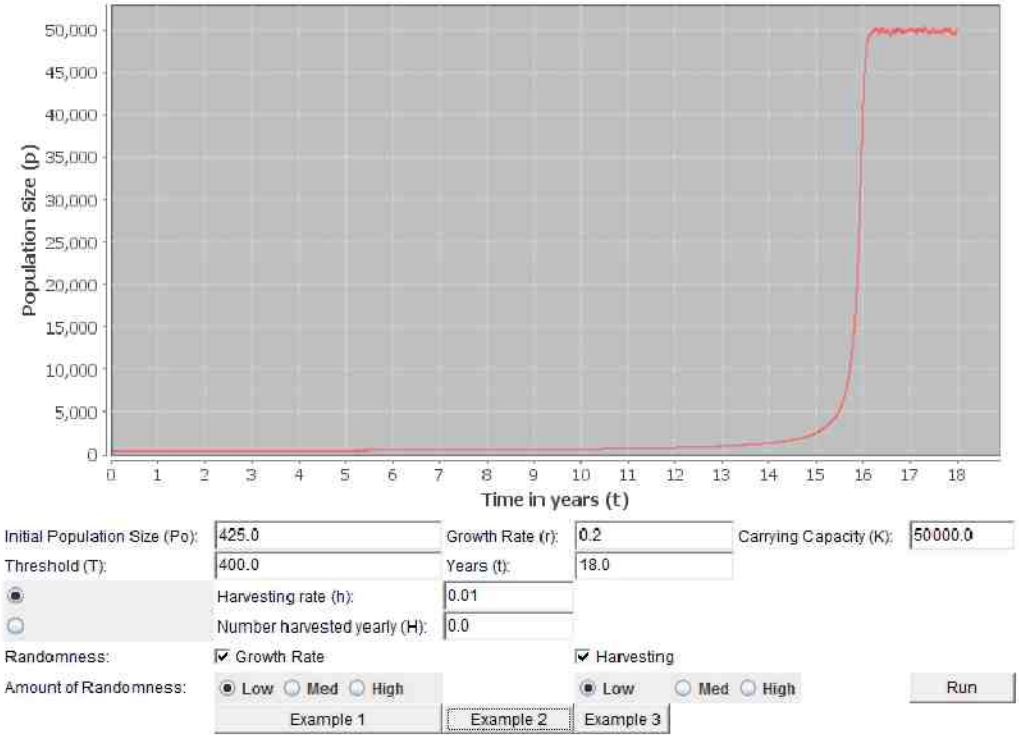
You may have noticed when you came to this page that something was quite different about this model, because the curve is no longer smooth. All the previous continuous models had smooth curves. Things changed in a predictable manner. This model incorporates randomness. Instead of things moving in a pre-determined way, they vary a little bit.

In this model, we can incorporate randomness in two places: in the growth rate and in harvesting. You can check the box for either or both of these to add in the randomness.

The equations for this model are the same as for the Logistic Growth Model with Harvesting [Section 2.4.5], with the addition of some random variables. The variables X_1 and X_2 are for the growth randomness, and the variables Y_1 and Y_2 are for the randomness associated with harvesting. The math behind these random variables is too complex for this website. If you are interested, you can read the Stochastic Differential Equations section in the paper about this website here [Section 7].

Example 1 [Figure 21] shows a population which is initially near its threshold. If there were no randomness, this population would survive and grow. With randomness, however, it is possible that the population will fall below its threshold and die out. This is why it is unsafe for a population to be near its threshold, because random events may drive it to extinction. Click Example 1 to see this. Each time you click, you will get a different graph because different random numbers are used. Click several times to see different possible outcomes. Example 2 [Figure 22] shows the same population with a few changes. The initial population is a little higher, which means the population has a better chance of survival. This population is also being harvested though, which lowers its chance of survival. It is dangerous to harvest a population which is near its threshold because, again, it may fall below its threshold and die out. Click Example 2 several times to see possible different scenarios.

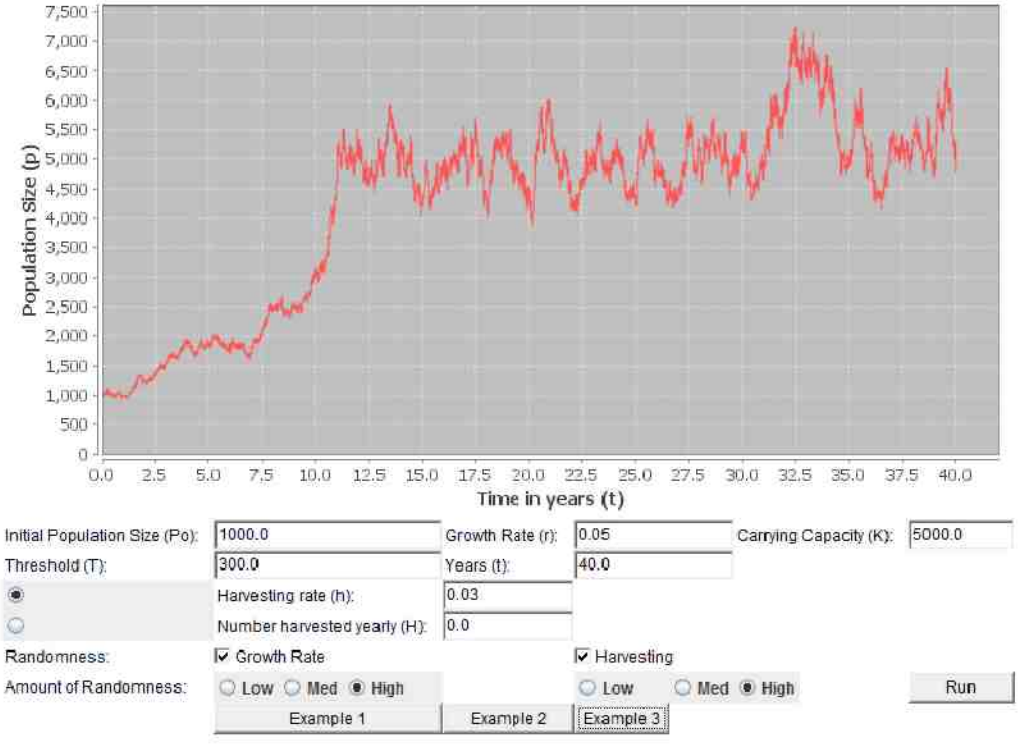
FIGURE 22. Randomness Model Applet (Example 2)



Below the check boxes for randomness, you can choose whether the amount of randomness is low, medium, or high. Click Example 3 [Figure 23]. This population is experiencing high randomness in growth rate and in harvesting. There is no telling what the population will look like at any given time. This is probably too much randomness for this population.

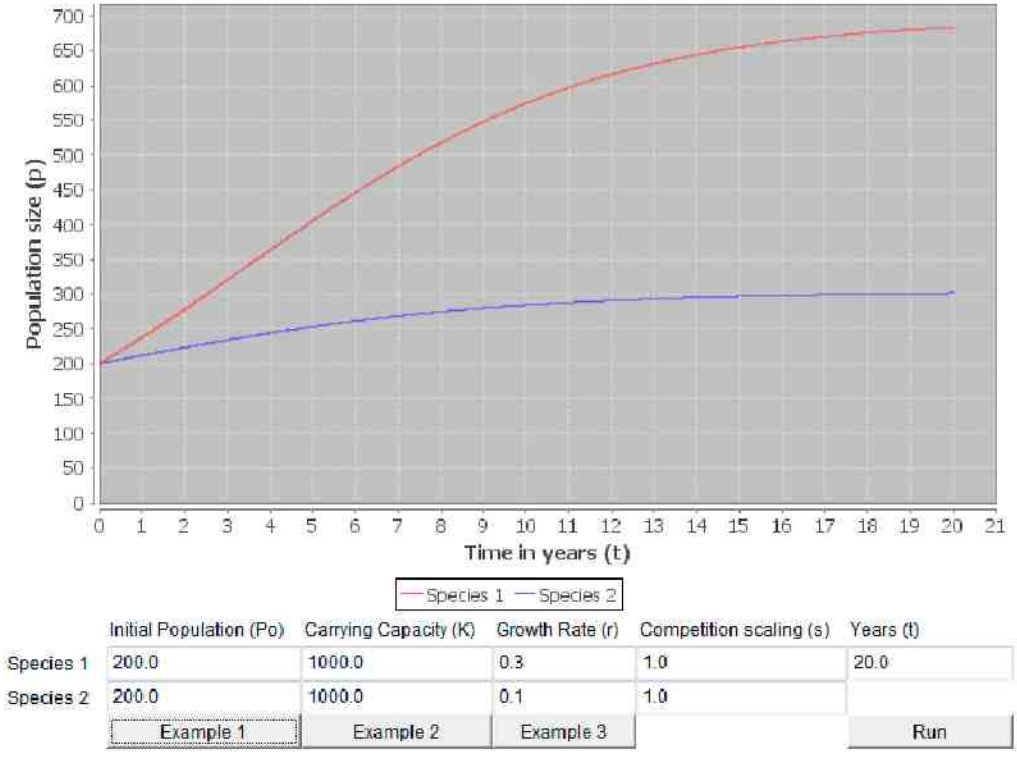
This concludes the single population models. We have seen that a population's size depends on its birth rate, death rate, carrying capacity, threshold, amount of harvesting, and random factors. However, it is rare that a population is isolated. In the multiple population models, we will see that a

FIGURE 23. Randomness Model Applet (Example 3)



population's size also depends on the size of prey, predator, and competitor populations. Click here [Section 2.1] to return to the main page, or click Next to move on to the multiple population models.

FIGURE 24. Competing Species Model Applet (Example 1)



2.4.7. *Competing Species Model.*

$$\frac{dp_1}{dt} \quad r_1 p_1 \left(1 - \frac{p_1 \mid s_1 * p_2}{K_1}\right)$$

$$\frac{dp_2}{dt} = r_2 p_2 \left(1 - \frac{s_2 * p_1 + p_2}{K_2}\right)$$

p population size, r growth rate, K carrying capacity,

s = competition scaling factor

No species exists in complete isolation. There are many models that show interactions between populations. In this model, we look at two species

competing for a limited resource, and see how this affects the size of both populations. In the Logistic Growth Model, we said that limited resources affect population growth. Each environment can support a limited number of animals. What happens when two species are using the same resources? Cows and rabbits both eat grass, so what if the grass is running low?

The Logistic Growth Model [Section 2.4.3] uses the differential equation below. (Click here [Section 2.2] if you do not know what a differential equation is.)

$$\frac{dp}{dt} \quad rp\left(1-\frac{p}{K}\right)$$

Here p is the population size, t is the time, $\frac{dp}{dt}$ is the rate of change in population size, r is the growth rate, and K is the carrying capacity. The growth rate tells us how quickly the population would grow in isolation with unlimited resources, and is the birth rate minus the death rate. The carrying capacity of a population is the number of animals in that population which can be sustained by the environment long-term.

Since we are dealing with two populations now, we will subscript the variables. Let p_1 be the size of Population 1 and r_1 the growth rate of Population 1. For Population 2 we have p_2 and r_2. Then, using the Logistic

Growth Model, our equation for Population 1 would be

$$\frac{dp_1}{dt} - r_1 p_1 \left(1 - \frac{p_1}{K}\right)$$

The carrying capacity piece of the equation, $1 - \frac{p_1}{K}$, works by limiting the growth of the population. If the population gets close to the carrying capacity, $\frac{p_1}{K}$ gets close to 1, so $1 - \frac{p_1}{K}$ gets close to zero. This means the change stops and the population size stabilizes. Now that there are two populations, this is the piece of the equation we will have to change.

As an example, let us have p_1 be a population of rabbits and p_2 be a population of cows. We determine that the environment can sustain either 1000 rabbits or 1000 cows. If both populations exist together, the environment can sustain 1000 animals in total. When the TOTAL number of animals gets near 1000, the rate of change of the rabbits will slow down to zero, the rate of change of the cows will also slow to zero, and the population sizes with stabilize. We can adjust the growth model to do this by changing $\frac{p_1}{K}$ to $\frac{p_1 + p_2}{K}$. Then the equations we get for the two populations are:

$$\frac{dp_1}{dt} = r_1 p_1 \left(1 - \frac{p_1 + p_2}{K}\right)$$

$$\frac{dp_2}{dt} - r_2 p_2 \left(1 - \frac{p_1 + p_2}{K}\right)$$

You can see a graph of this example by clicking Example 1 [Figure 24]. (Just ignore the s for now.) In this example, the rabbits have a higher growth rate than the cows, so when it levels off there are more rabbits than cows. The total number of animals at the end is 1000. You can experiment with the growth rates and the initial populations to see different outcomes. One population may even become extinct.

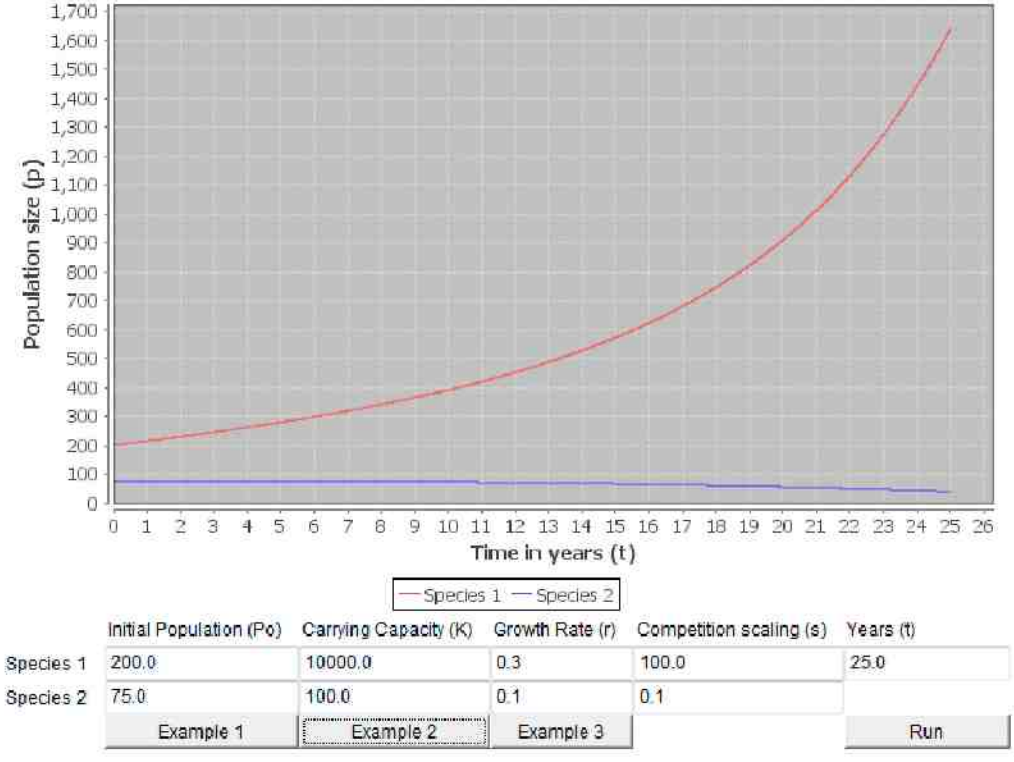
The problem with this model is it assumes that the carrying capacity is the same for both populations. It assumes that the environment can sustain 1000 rabbits or 1000 cows, and both eat an equal amount of grass. We can modify the model by putting in competition scaling factors, s_1 and s_2.

$$\frac{dp_1}{dt} = r_1 p_1 \left(1 - \frac{p_1 \mid s_1 * p_2}{K_1}\right)$$

$$\frac{dp_2}{dt} \quad r_2 p_2 \left(1 - \frac{s_2 * p_1 + p_2}{K_2}\right)$$

The program above uses these equations. Here, each species has its own carrying capacity, K or K_2. The environment may be able to support 10,000 rabbits (K = 10000) but only 100 cows (K_2 = 100). The variables s and s_2 are competition scaling factors. If a cow eats 100 times as much as a bunny, s is 100 and s_2 is 1/100 0.01. This modified example is shown in Example 2 [Figure 25]. You can change the initial populations again

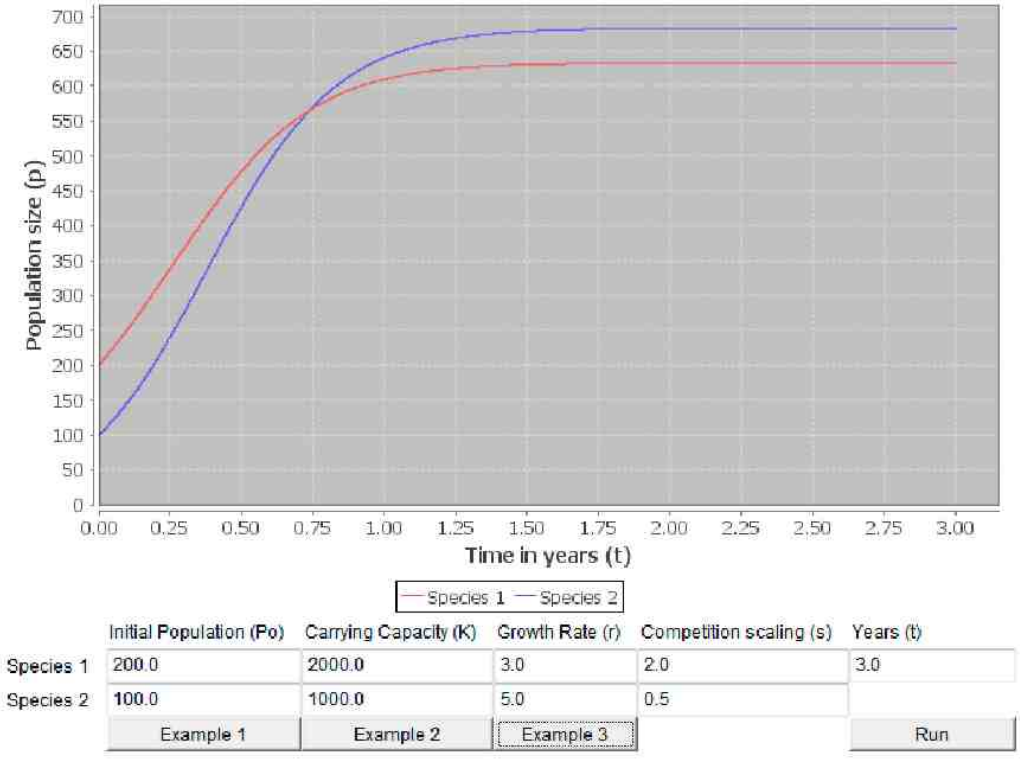
FIGURE 25. Competing Species Model Applet (Example 2)



and see that there are different possible outcomes. If you lower the initial number of cows, the rabbits will take over, and there will never be many cows.

Sometimes it may not be obvious which species will end up dominating. Example 3 [Figure 26] shows two species of birds competing for nesting space. The first birds have a higher initial population and a higher carrying capacity, so for a while there are more of them. However, after a few years the second bird overtakes the first, because the second birds have a higher growth rate.

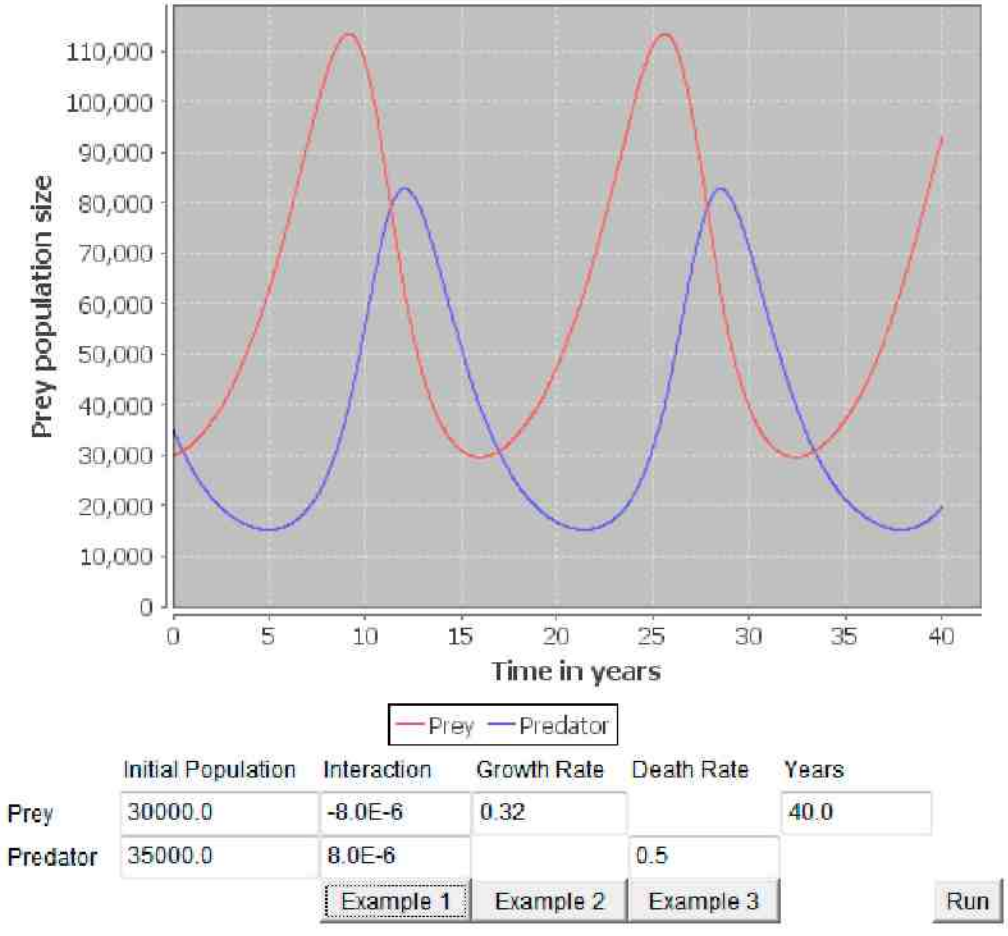
FIGURE 26. Competing Species Model Applet (Example 3)



Keep in mind that other factors can affect how species compete. For example, one may be more aggressive than the other. This model only takes into account the factors above.

When you are done with this page you can return home [Section 2.1] or you can move on to the predator-prey model.

FIGURE 27. Predator-Prey Model Applet (Example 1)



2.4.8. *Lotka-Volterra Predator-Prey Model.*

$$\frac{dx}{dt} \quad x(a - by)$$

$$\frac{dy}{dt} = -y(d - cx)$$

x prey population size, y predator population size, a prey growth rate, d predator death rate, b and c are interaction terms

When a predator population eats members of a prey population, the size of each population depends on the size of the other. The size of the prey population decreases as they are eaten (by the predators), and the size of the predator population decreases when they do not have enough food (prey). Many predator-prey models have been proposed, and this page looks at the first one to be accepted, the Lotka-Volterra Predator-Prey Model.

The Lotka-Volterra model was created independently by two mathematicians in the 1920's. Alfred J. Lotka published it in 1925, and Vito Volterra in 1926 [11]. All mathematical models make assumptions that result in a simplification of the natural world. This model assumes that the prey population is eaten only by a single predator population, and that the predator population eats the prey population exclusively. It also assumes that the prey have unlimited food, and that there are not any other interactions with these populations.

Predator and prey populations often follow a cycle. Suppose that both population sizes start small. Then the prey population increases, as populations tend to do. Since there are more prey, the predators eat better and reproduce more. This results in an increase of the predator population.

Then there are too many predators, and they eat too many of the prey. This causes the prey population to decrease. A decrease in prey means the predator population will decline because they do not have enough food. This brings us back to the beginning of the cycle, and the prey population will start increasing again since there are less predators.

In this model, the predator and the prey have distinct differential equations. (If you do not know what a differential equation is, click here [Section 2.2].) We will distinguish between the two populations by calling the prey population x and the predator population y. Let us start by looking at the prey equation. Most of the single population models on this website are based off the Logistic Growth Model [Section 2.4.3], but this model is actually based off the Continuous Exponential Model [Section 2.4.2]. If there were no predators, and if we let a be the growth rate of the prey, this model gives us:

$$\frac{dx}{dt} \quad ax$$

Here $\frac{dx}{dt}$ is the rate of change of the prey population. The exponential model assumes that there are no predators. When predators are present, they will cause a decrease in the prey population's rate of change. The Lotka-Volterra model assumes that this effect will be proportional to both the size of the predator population and the prey population. This means

when there are more predators, more prey will be eaten. Moreover, when there are more prey, more prey will be eaten, because the predators will have more food available to eat. Let b be the predation rate, which relates how many prey are eaten to the number of predators and prey. We will modify the equation above by subtracting off bxy. The equation we get for the prey is

$$\frac{dx}{dt} = ax - bxy$$

which can be factored to give

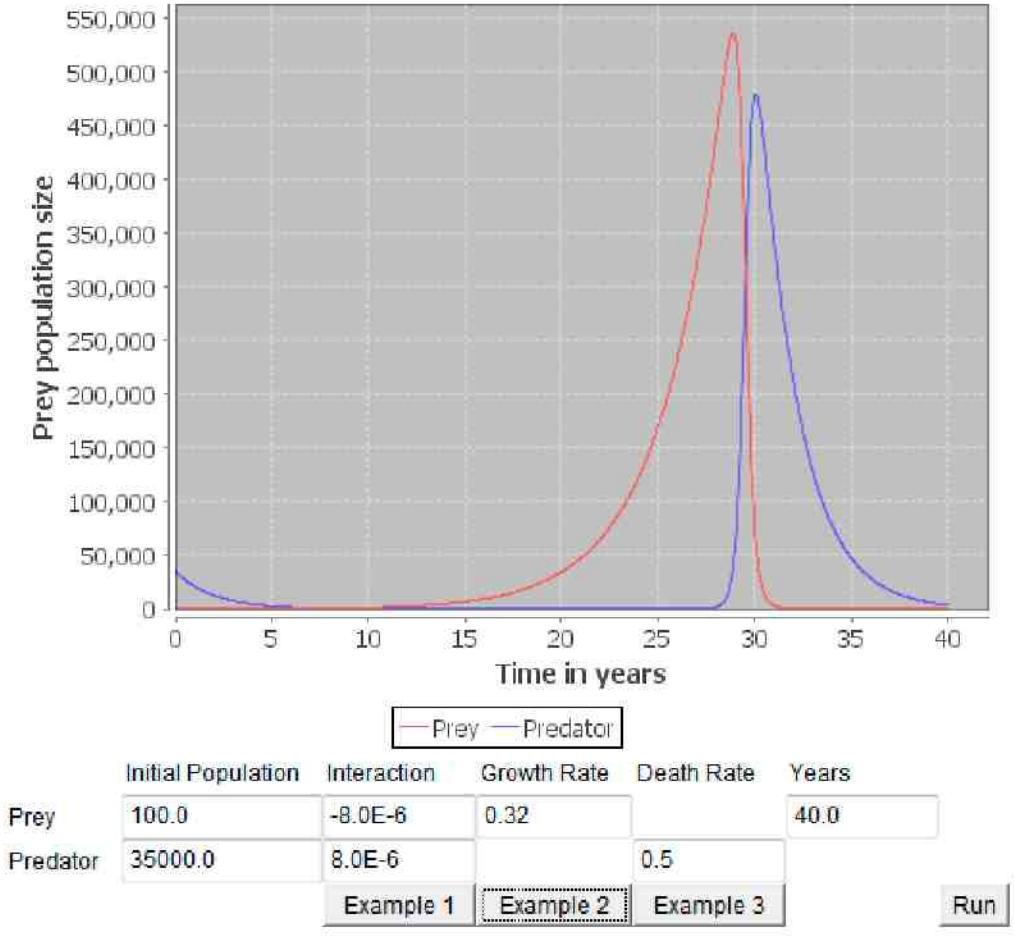
$$\frac{dx}{dt} = x(a - by)$$

This is the Lotka-Volterra prey equation.

Next we have to find the predator equation. We start by making an equation that does not involve the prey. What does the predator population rate of change look like if there are no prey? They would simply die out. If we let d be the death rate of the predators in the absence of prey and restrict d to a positive number, the equation we get is

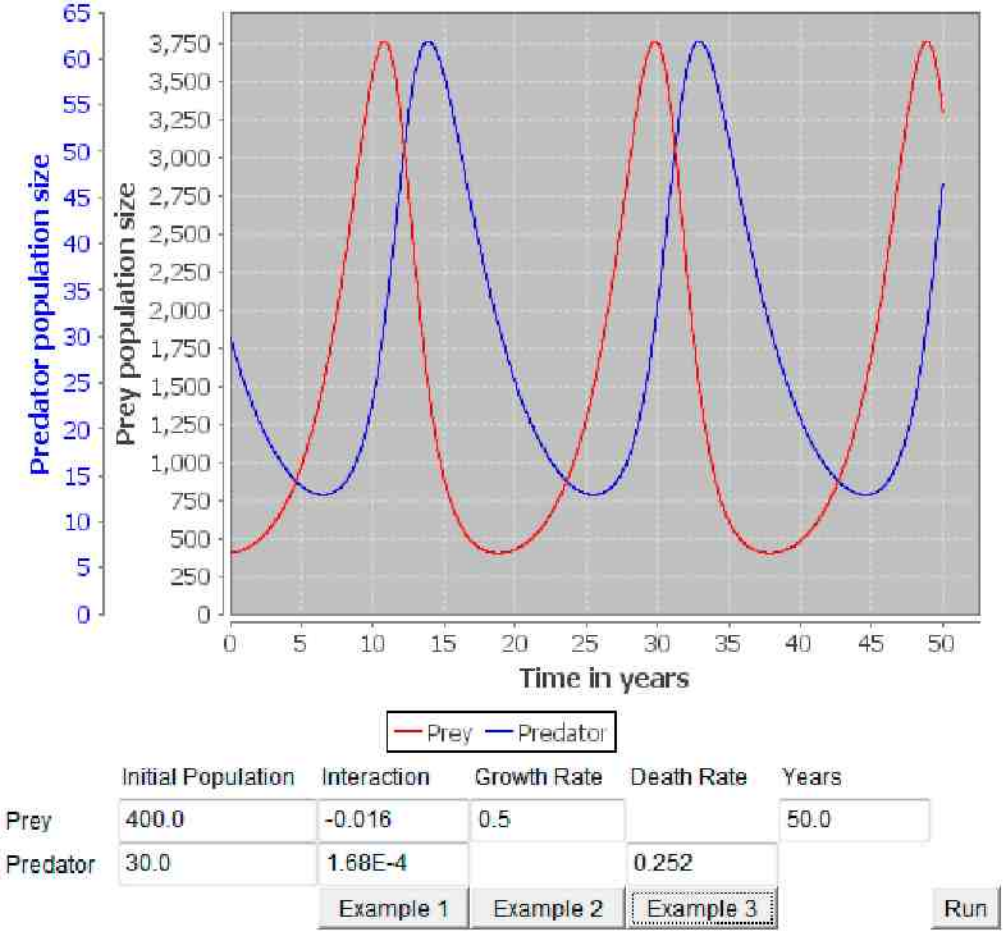
$$\frac{dy}{dt} - -dy$$

FIGURE 28. Predator-Prey Model Applet (Example 2)



When there are prey available, they will have a positive effect on the rate of change of the predator population. The Lotka-Volterra model assumes that this effect will be proportional to the number of predators and also the number of prey. When there are more prey, the predator population will increase. When there are more predators, the predator population will increase more because they produce more offspring. Let c be the constant

FIGURE 29. Predator-Prey Model Applet (Example 3)



that relates the increase in predators with the number of predators and prey. Then the change will be cxy. Our equation becomes

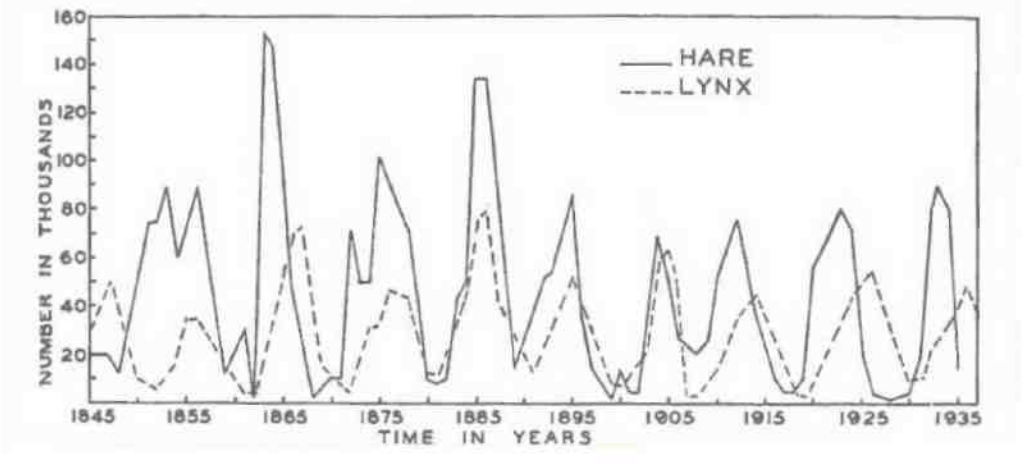
$$\frac{dy}{dt} \quad -dy + cxy$$

Which factors to

$$\frac{dy}{dt} \quad -y(d - cx)$$

This is the Lotka-Volterra prey equation. Click Example 1 [Figure 27] to see a situation where hares are eaten by lynx. Like all mathematical models, this is an approximation of the real world. The real data[13] for the lynx and the hares is in the image below [Figure 30].

FIGURE 30. Graph of the Lynx and Hare Data

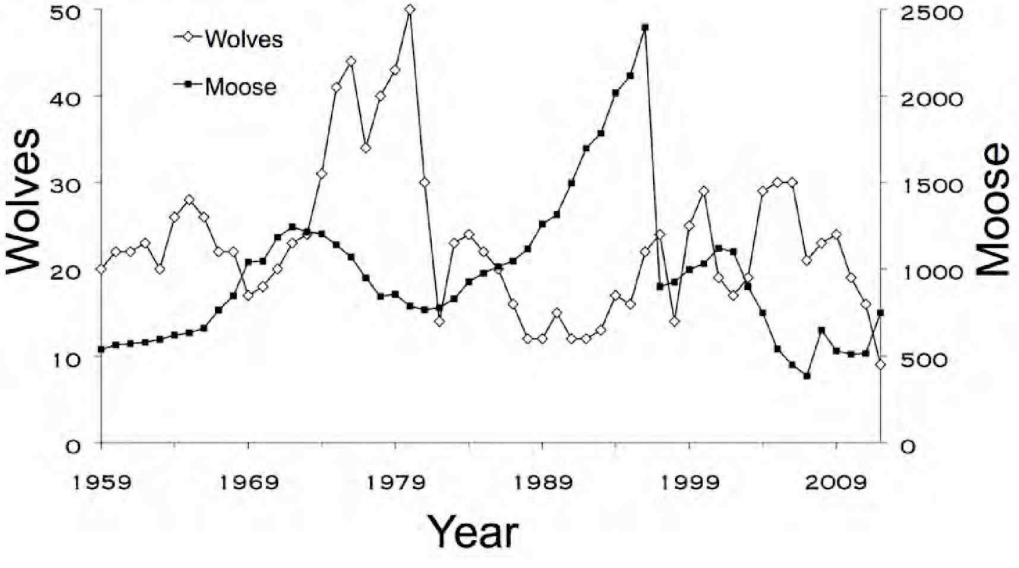


The major problem with this model can be illustrated in the following example. Suppose c 2 and d 0.1 and there are 10 predators. If there are 100 prey, the rate of change of the predator population is -2(10)+0.1(100)(10) 80. If there are 100,000 prey, the change is -2(10)+0.1(100000)(10) 99,980. The rate of change of the predators was multiplied by about 1250. This does not make much sense. There should be a limit to the number of new predators that can be born, no matter how much food is available to them. Another problem this model has is it uses no threshold. If enough prey are eaten they should die out, but they do not. Click Example 2 [Figure 28] to

see an unrealistic situation with the lynx and hares. As mentioned above, there are many more predator-prey models that improve upon this one, and some of them correct this type of situation.

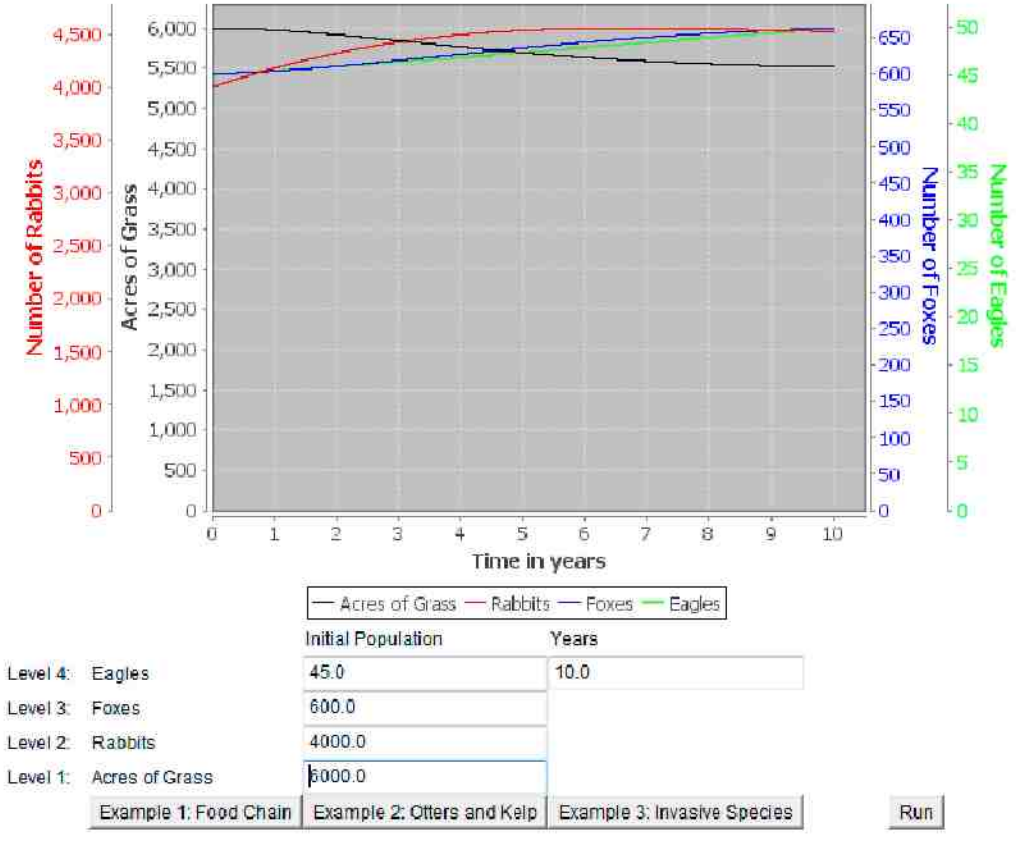
Example 3 [Figure 29] shows wolves eating moose. This is another example that has been taken from real life, from Isle Royale, an island in the Great Lakes. The real data[22] can be seen in the image below [Figure 31].

FIGURE 31. Graph of the Moose and Wolf Data



Next you can move to the last model on this website, Balanced Ecosystems.

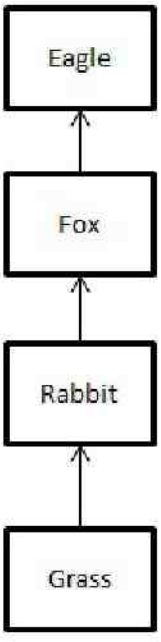
FIGURE 32. Balanced Ecosystems Model Applet (Example 1)



2.4.9. *Balanced Ecosystems.* In the Lotka-Volterra Model [Section 2.4.8], we saw examples of one species being a predator and another being prey. However, one animal's predator can be another animal's prey. For example, a snake eats a bug and is eaten by a hawk. This model looks at several species in a food web. A change to one population may affect all the others, in a small or large way. This page contains three examples of food webs that are in balance, but the balance can be upset. The equations are a little more complicated so we have left them off this page. If you are interested,

you can read the "Equations for the Balanced Ecosystems Applet" section in the paper about this website here [Section 5].

Example 1: Food Chain

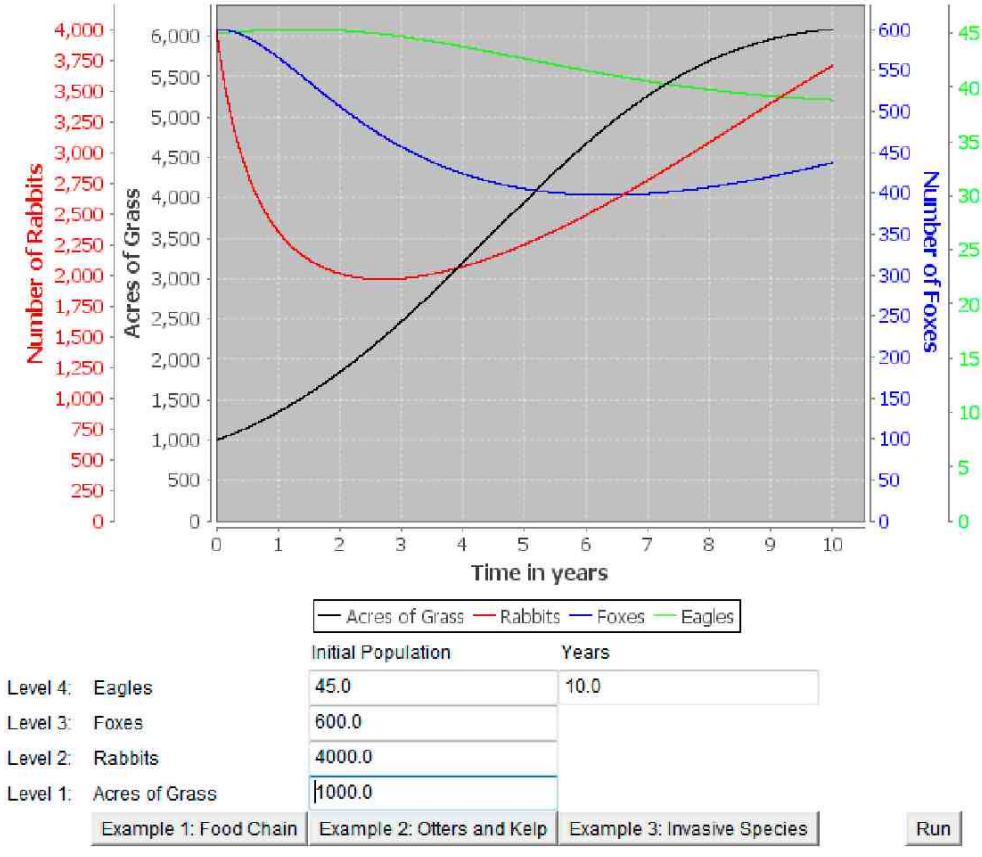


This first example [Figure 32] shows a simple food chain. On the bottom is grass. Rabbits eat the grass, foxes eat the rabbits. On top are eagles, which eat the foxes. Right now, they are in a good balance. Take a minute and answer this question: What do you think will happen if we kill most of the grass?

If the grass dies, the rabbits will die because they have no food. When the rabbits die, the foxes will die, and then the eagles will die. By killing a species at the bottom of the chain, we affect the species all the way up the chain. Now try changing the amount of grass to 1000 acres or less. [See Figure 33.]

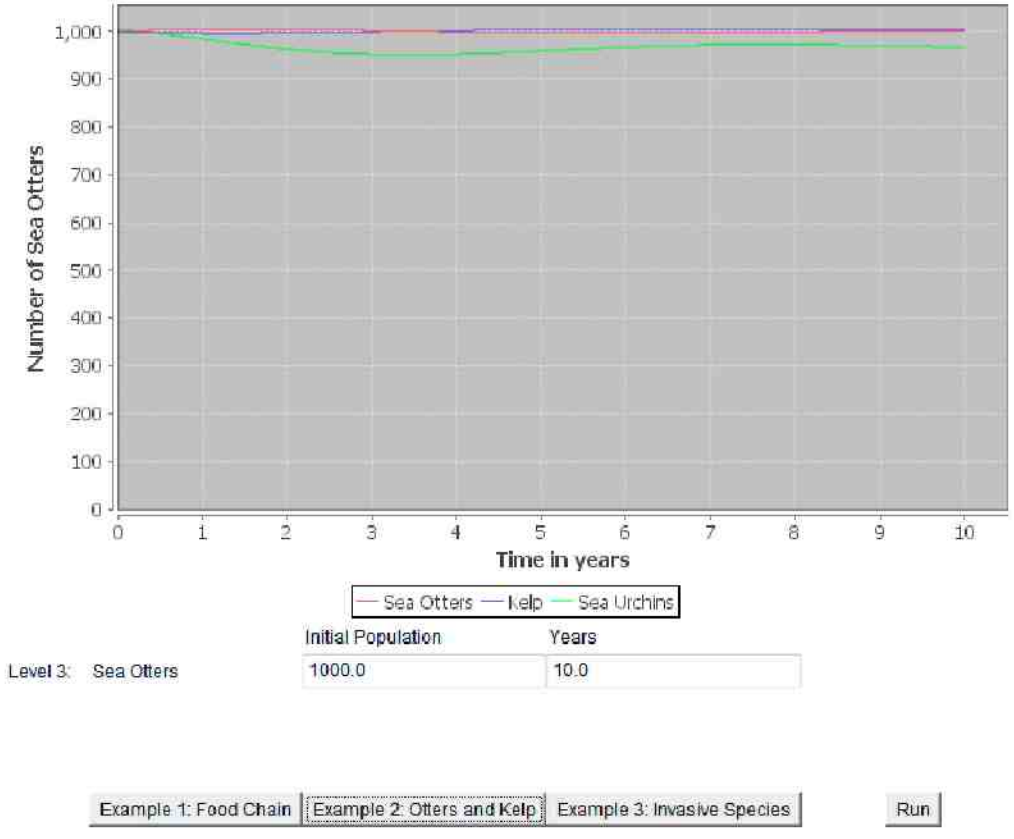
Remember that all population models are imperfect. Each model works well for specific purposes. In this model, after a while the populations will all recover. Change the number of years to 100, and you will see that the populations always reach a balance. There is no threshold built into this model, so no matter how far the populations fall, they will eventually come back up. (See the Logistic Growth Model with a Threshold [Section 2.4.4]

FIGURE 33. Balanced Ecosystems Model Applet (Example 1 Changed)



if you are interested in what a threshold is.) In reality, it is possible that if too many of one species died, they would end up going extinct. This model also does not take into account any randomness. (To see a model with randomness, click here [Section 2.4.6].)

FIGURE 34. Balanced Ecosystems Model Applet (Example 2)

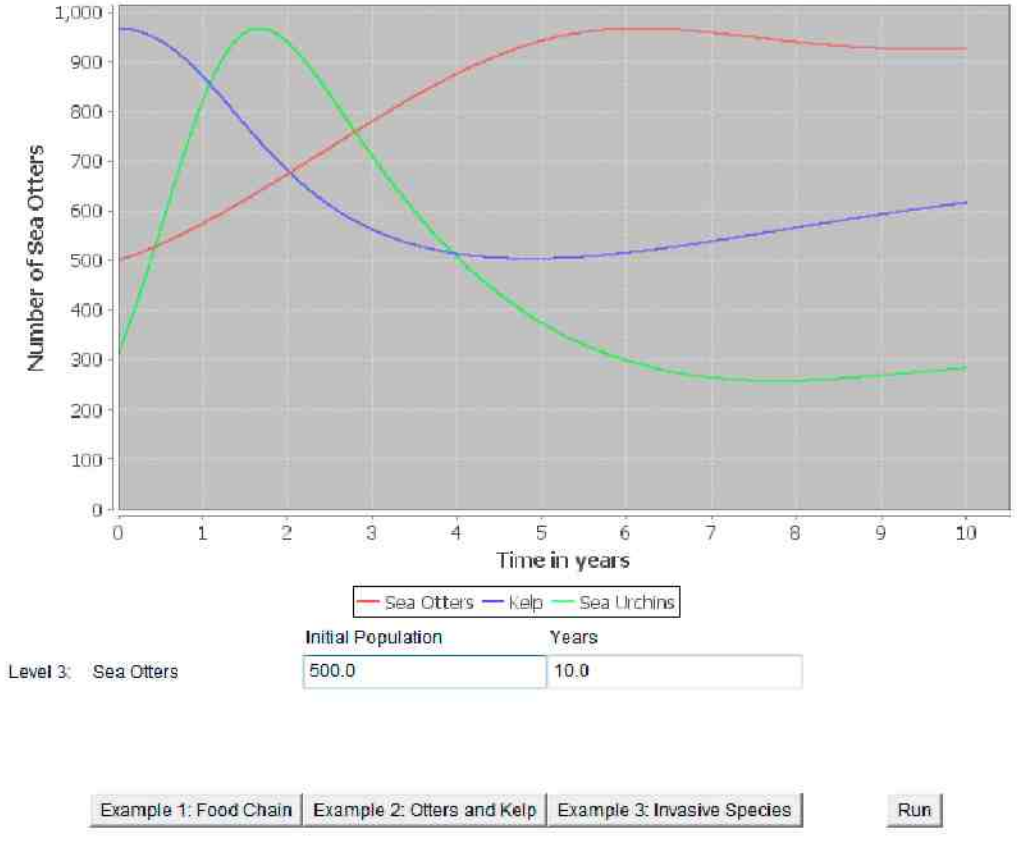


Example 2: Otters and Kelp

Our next example [Figure 34] is another simple food chain. Sea otters eat sea urchins, which eat kelp. Take a minute and answer this question: What do you think will happen if hunters or killer whales kill many of the sea otters?

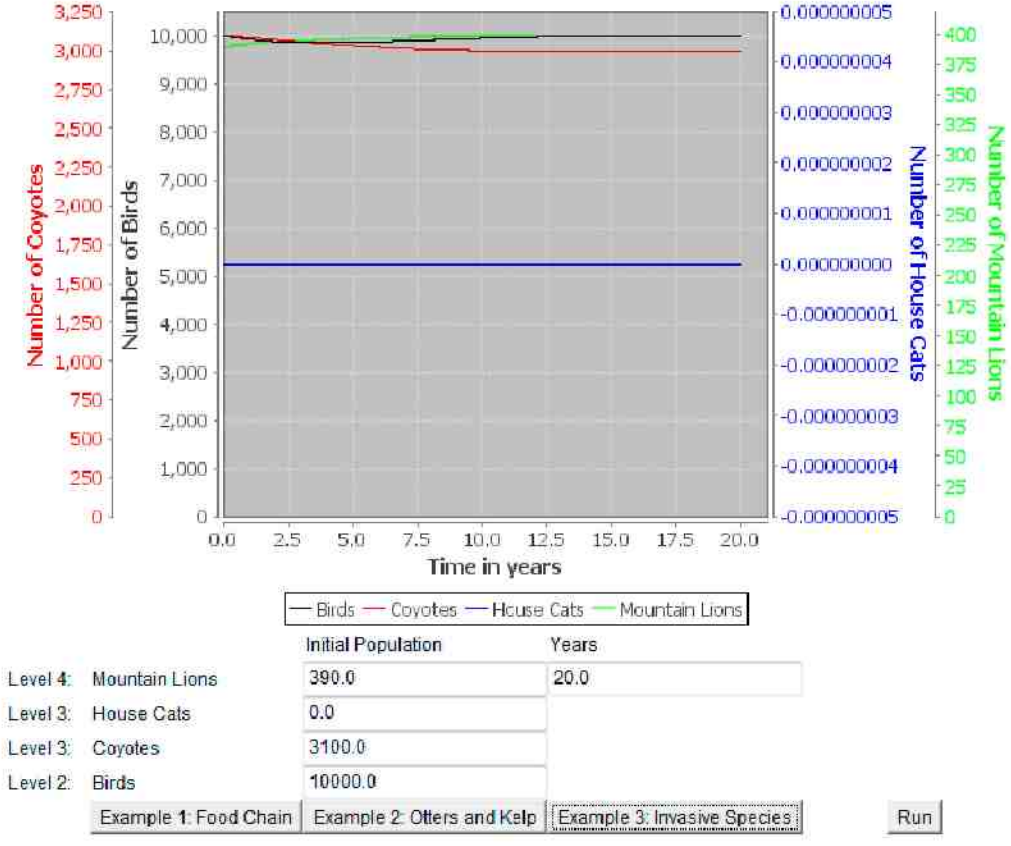
If we lose many sea otters, the sea urchin population will increase. When there are more sea urchins, they will eat more kelp, so the amount of kelp will decrease. By hunting the sea

FIGURE 35. Balanced Ecosystems Model Applet (Example 2 Changed)

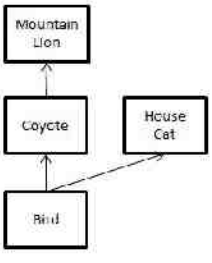


otters, we threw off the entire food chain. Change the number of sea otters to 500 in the program above. [See Figure 35.]

FIGURE 36. Balanced Ecosystems Model Applet (Example 3)

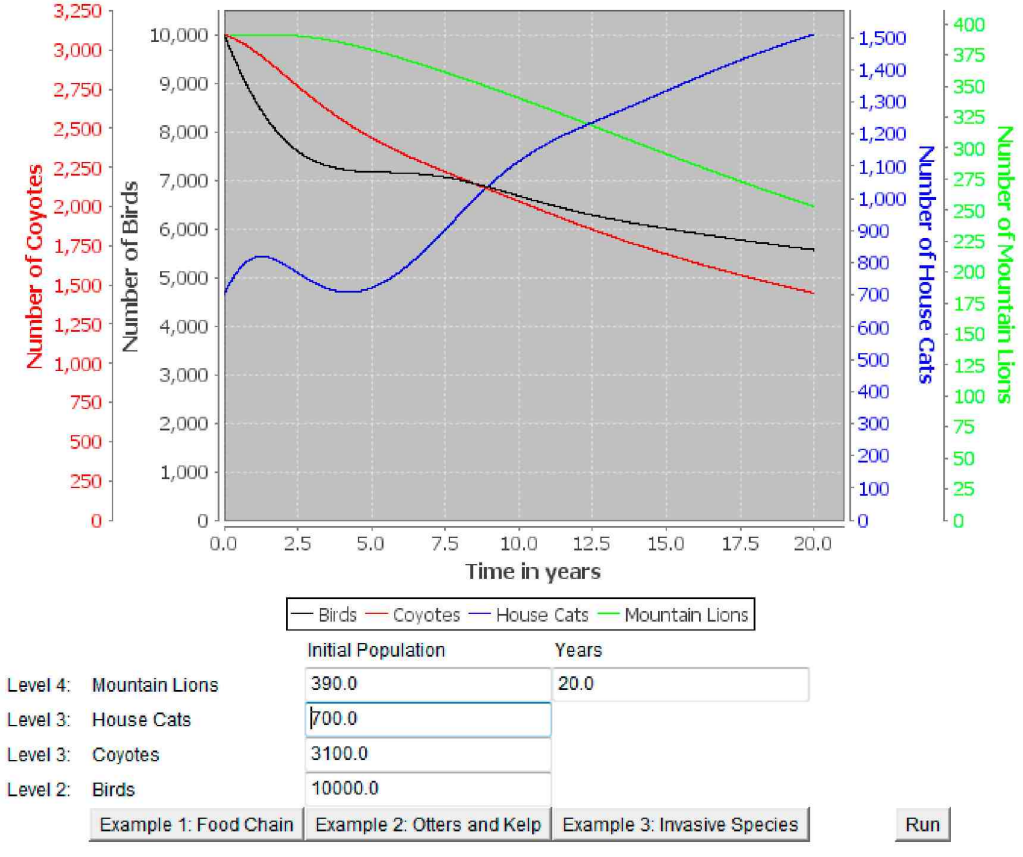


Example 3: Invasive Species



The third example [Figure 36] has a different food chain. Mountain lions eat coyotes, which eat birds. What will happen if we introduce another species into the situation? House cats are released by people who cannot take care of them, or they escape, or are born from stray cats. The cats eat the birds, which were food for coyotes. The coyotes will not find

FIGURE 37. Balanced Ecosystems Model Applet (Example 3 Changed)



enough food, which leads to the mountain lions not having enough food.

Right now Example 3 shows a balanced system without cats. Change the number of cats and watch what happens. [See Figure 37.]

This is the last model we have so far. You can return to the homepage [Section 2.1] or send us feedback.

3. Website Technical Information

The website is written with HTML and CSS (Cascading Style Sheets). There is one HTML file for each page. There is one style sheet for the homepage, and one style sheet shared by all the model pages.

The Java files are bundled together into one file, called a JAR file (Java Archive file). When the user visits one of the pages containing an applet, the JAR file is automatically downloaded. The first page they visit may take a few seconds to load, but any other pages they visit will load more quickly because they have already downloaded the file.

There are various equations throughout the website. The website uses MathJax to display these. MathJax is "an open source JavaScript display engine for mathematics that works in all modern browsers."[17] To use MathJax, the following code is contained within the <head> block of the HTML file for each page.

```
<script type="text/x-mathjax-config">MathJax.Hub.Config
({tex2jax: {inlineMath: [['$','$'], ['\\(','\\)']]},
"HTML-CSS": {preferredFont: "TeX"}});</script>
<script type="text/javascript' src="http://cdn.mathjax.
org/mathjax/latest/MathJax.js?config=TeX-AMS-
MML_HTMLorMML"></script>
```

Wherever we desire math images, we write LaTeX math code inside of dollar signs. After the browser has loaded a page, it runs the script from the web address above. The script looks for the text inside dollar signs, and the MathJax Content Delivery Network returns images to put in place of the math code. An example from the website is:

> If \$KS is very large compared with \$p\$, \$\frac{p}{K}\$ will be close to 0.

This will turn into

> If K is very large compared with p, $\frac{p}{K}$ will be close to 0.

Single dollar signs signify inline math, as above, while double dollar signs signify displayed math, as below.

$$\lim_{K\to\infty} rp\left(1-\frac{p}{K}\right) = rp(1-0) = rp$$

Because people usually write dollar signs to signify dollars and not math code, the default setting for MathJax is not to use a single dollar sign to indicate inline math. However, we did not anticipate using dollar signs in the text of this website, so I changed the setting to allow dollar signs for inline math. That was done using the piece above that says
"inlineMath: [['\$','\$'] ."

4. Programming

4.1. **Program Overview.** We wanted the programs and the website to be compatible with as many computers as possible. For this reason, I chose to write the programs as Java applets. Programs written in Java work on virtually any platform, and over a billion computers run Java[19]. Applets run through a web browser, and do not require the user to manually download or install anything new.

The programs were originally written in Java 7. However, the computers at CSUCI on which the website was tested could not run it, because they only had Java 6. Typically having a different version would not make much difference, but Java 7 introduced the ability to "switch on Strings," so wherever the program made use of this it was not compatible. To correct this the program was downgraded to Java 6, and the "String switches" were changed to "nested if statements" with "String comparisons."

Five Java files were written for this project. "SinglePopulation java" is the applet for all the single population models. Because each single population model builds upon the previous one, it was logical to combine them into a single program. Each single population webpage passes a parameter, "model," to the program to let it know which model to use. "Model" is a keyword of the title of the model. For example, "Logistic Growth with

Harvesting" is simply called "Harvesting." Each model also has a number assigned to it in sequential order, so that each model contains all the features of the models with lower numbers. This is useful in the program because it allows us to use "switch" statements and "greater-than" or "less-than" comparisons. Table 1 shows the numbers and keywords assigned to each model, as well what new features each model includes.

The single populations models use one other file, RungeKutta java. This file is also used by the multiple population models. This Java class implements the Runge-Kutta Method of approximation for systems of differential equations. RungeKutta java is discussed in Section 4.3.

The multiple population models each have their own files. These are CompetingSpecies java, PredatorPrey java, and BalancedEcosystems java.

TABLE 1. Levels of single population models

Number	Model Name	Keyword	Additional Features
1	Discrete Exponential	Discrete	Initial Population, Growth Rate, Years
2	Continuous Exponential	Exponential	None
3	Logistic Growth	Logistic	Carrying Capacity
4	Logistic Growth with a Threshold	Threshold	Threshold
5	Logistic Growth with Harvesting	Harvesting	Harvesting Rate, Harvesting Amount, Radio Buttons
6	Logistic Growth with Harvesting and Randomness	Stochastic	Radomness High, Medium, Low Buttons; Checkboxes

The chart images displayed in the programs are created by JFreeChart, which is a free Java package available for download online[18]. It was only necessary to be downloaded by the developer; the user gets it automatically, since it is bundled into the JAR file. The population programs generate the list of points using RungeKutta java, and then JFreeChart takes those points and generates a graph.

4.2. **Program Methods.** The following sections contain brief descriptions of the methods contained in SinglePopulation java. The multiple population files are very similar. For the complete code of SinglePopulation java, see Appendix A. To see how they are connected, see "Program Flow" in Section 4.4.

4.2.1. *void init().* This is the initializing method. It looks for the model parameter passed in from the webpage. Then it sets the model number, according to Table 1. It finishes by calling the method initialDraw().

4.2.2. *void initialDraw().* This method draws everything on the screen. It uses if statements to draw only the necessary components. For example, if the model number is greater than 3, it adds the threshold box and label. All the models after number 3 use a threshold, but the models before do not. (See this in Table 1.) This method calls setExample(), setVariables(),

setTextFields(), createDataset(), and createChart(). After it finishes, the program waits for user input.

4.2.3. *void redraw(boolean).* This redraws the chart and the text fields. This is called by one of the other methods based on user input. If it is passed a value of true, this means it should redraw based on the user input. It does some error-checking on the user input, and changes the error message as appropriate. For example, if the user enters a letter or a negative number for the population size, the message will be set to "Initial population must be a positive number." This is displayed in red text at the bottom of the applet. If the method is passed a value of false, it calls setExample() to change the class variables. This method calls setTextFields(), createDataset(), and createChart().

4.2.4. *XYDataset createDataset().* This is the mathematical part of the program. For the discrete and random models, the data points are generated in this method. For the other models, we use the current values in the class variables to change the coefficients of the differential equation. Each model uses a different type of differential equation. A RungeKutta object is created with the coefficients and some other information. After is it created, the RungeKutta object will hold the data points.

4.2.5. *void setExample().* This method loads the appropriate example into the class variables using setVariables().

4.2.6. *void setVariables(double, double, double, double, double, double, double).* The class variables are set to the numbers passed into this method. This is just a shortcut method, so that we do not have to write out all the assignments every time we want to change the variables.

4.2.7. *void setTextFields().* This method turns the numeric variables into text, and sets the text in the appropriate TextFields. For example, if the variable "years" has a value of 10, this has to be converted into a String, '10'. The yearsTextField is then assigned the String '10', and the screen displays a 10.

4.2.8. *JFreeChart createChart().* This method creates a chart using JFreeChart and the dataset generated by createDataSet().

4.2.9. *void actionPerformed().* This method is called when the user clicks on one of the buttons. If one of the example buttons was selected, exampleNumber is changed to the correct number, and redraw() is called and given a parameter of false. If the "Run" button was selected, redraw() is called and given a parameter of true.

4.2.10. *void keyPressed().* This method is called when the user presses "Enter" on the keyboard while one of the buttons is selected. It is essentially the same as actionPerformed(), and calls redraw() if appropriate.

4.2.11. *void mouseClicked().* This method is called when the user clicks on harvestProportionText or harvestAmountText. It selects the radio button associated with the text box. The purpose of this is to avoid an error caused by the user forgetting to select the appropriate radio button.

4.2.12. *void keyTyped().* This method is called when the user types in harvestProportionText or harvestAmountText. It is the same as mouseClicked().

4.2.13. *void focusGained().* This method selects the text in a text box when a user clicks or tabs to it. This way the user does not have to delete the text that was in there. When they start typing the new information, the old information will be automatically deleted.

4.3. **RungeKutta Class.** This class performs the Runge-Kutta method explained in Section 6. When one of the model classes creates a RungeKutta object, it gives the following parameters: coefficients of the equations, initial values, start of the range, end of the range, number of points, number of equations, and model. By the time RungeKutta is finished creating the object, it contains all the points that will be plotted. The code for RungeKutta java is contained in Appendix B

4.4. **Program Flow.** The figures below depict the program flow. Methods are represented by boxes, and call the methods below them, from left to right.

FIGURE 38. Program flow on opening

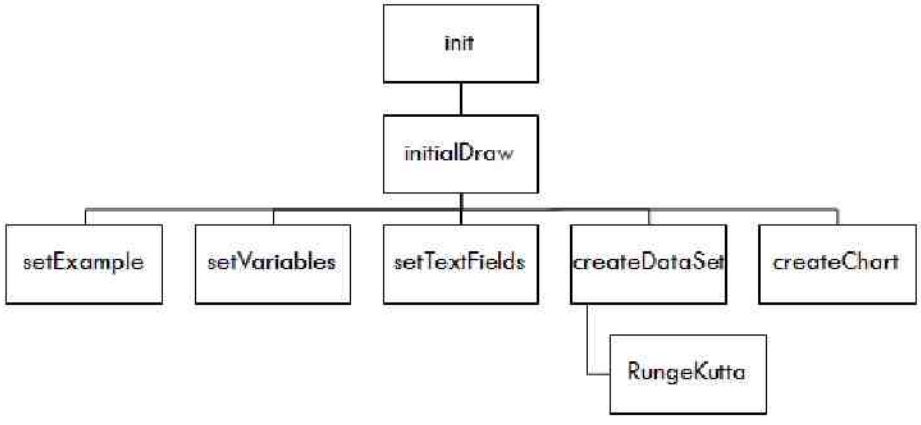


FIGURE 39. Program flow when user selects Run button

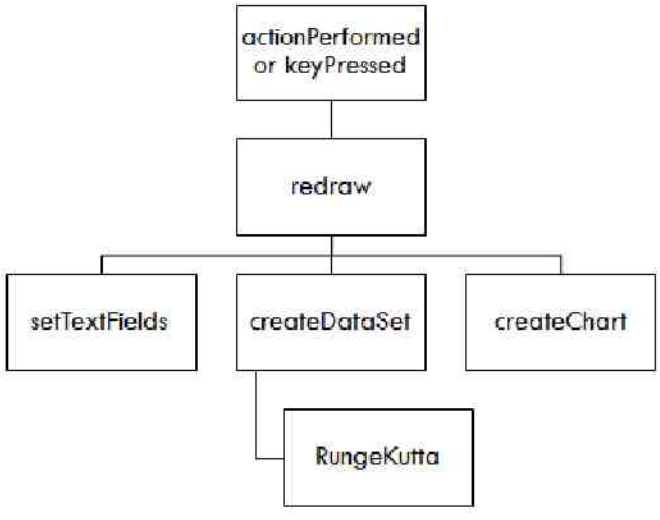
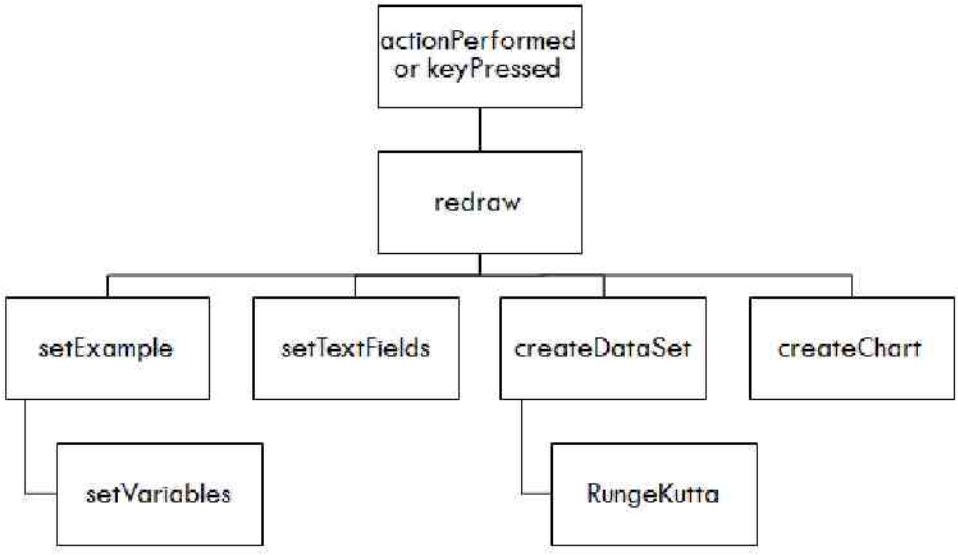


FIGURE 40. Program flow when user selects Example button



5. Equations for the Balanced Ecosystems Applet

The Balanced Ecosystems webpage does not show the equations used, so we will look at them here. The applet contains three different examples, each with three or four populations. We wanted a model in which one species could be predator, prey, and competitor all at once. We created the equations by combining other models.

For the predator-prey interaction, the model we used on the website was the Lotka-Volterra Model. However, this model causes exponential growth in the prey in the absence of predators. Also, the predator population grows too rapidly when there are large numbers of prey. Instead of the Lotka-Volterra Model, we will start with the model introduced by P. H. Leslie[3].

Let N be the prey population and let P be the predator population. The model is:

$$\frac{dN}{dt} = aN\left(1 - \frac{N}{K}\right) - bNP$$

$$\frac{dP}{dt} = cP\left(1 - \frac{eP}{N}\right)$$

In order to make a comparison with the carrying capacity, we can choose to rewrite this by letting $f \quad 1/e$.

(2) $$\frac{dN}{dt} \quad aN\left(1 - \frac{N}{K}\right) - bNP$$

(3) $$\frac{dP}{dt} = cP\left(1 - \frac{P}{fN}\right)$$

In Equation 2, K is the carrying capacity of the prey. In Equation 3, fN is essentially the carrying capacity of the predator. The number of predators is limited by the number of prey. We will use this idea to extend the model. For a series of populations $x_1, x_2, \ldots, x_n$, where x_i eats x_{i-1} for all $i > 1$, we have the following differential equation to describe each population with $1 < i < n$.

(4) $$\frac{dx_i}{dt} = r_i x_i\left(1 - \frac{x_i}{f_i x_{i-1}}\right) - b_i x_i x_{x\,|\,1}$$

We use Equation 4 in the Balanced Ecosystems Applet, Examples 1 and 2. The third example has competing species as well. To incorporate this, we combine Equation 4 with the Competing Species Model [Section 2.4.7]. If each population x_i has at most one competitor x_s with competition scaling factor s_i, prey x_f, and predator x_b, the model we get is:

$$\frac{dx_i}{dt} = r_i x_i \left(1 - \frac{x_i + s_i x_s}{j_i x_f}\right) - b_i x_i x_b$$

If a species has no competitor or no predator, we simply remove that piece of the equation. If a species has more than one competitor, we put another "b" term at the end. As with all models it has its downsides, but it is sufficient for our purpose of showing interactions between species.

6. Runge-Kutta Approximation Method

The Runge-Kutta methods are processes used to approximate solutions to systems of differential equations. They were invented by C. Runge and M. W. Kutta around 1900 [29]. This program uses the fourth-order Runge-Kutta Method. The procedure is in Section 6.1, and an explanation follows in Section 6.2. Section 6.3 describes a slight modification that was made to the procedure for this project.

6.1. **Runge-Kutta Procedure.**

Runge-Kutta Method for Systems of Differential Equations[6]

To approximate the solution of the mth-order system of first-order initial-value problems

$$u_j' = f_j(t, u_1, u_2, \ldots, u_m),\ a \le t \le b,\ \text{with}\ u_j(a) = \alpha_j,$$

for $j = 1, 2, \ldots, m$ at $(N + 1)$ equally spaced numbers in the interval $[a, b]$:

INPUT endpoints a, b; number of equations m; integer N; initial conditions $\alpha_1, \ldots, \alpha_m$.

OUTPUT approximations w_j to $u_j(t)$ at the $(N + 1)$ values of t.

Step 1. Set $h = (b - a)/N$; $t = a$.

Step 2. For $j = 1, 2, \ldots, m$ set $u_j = \alpha_j$.

Step 3. OUTPUT $(t, w_1, w_2, \ldots, w_m)$.

Step 4. For $i = 1, 2, \ldots, N$ do steps 5–11.

Step 5. For $j = 1, 2, \ldots m$ set $k_{1,j} = hf_j(t, w_1, w_2, \ldots w_m)$.

Step 6. For $j = 1, 2, \ldots m$ set $k_{2,j} = hf_j(t + \frac{h}{2}, w_1 + \frac{1}{2}k_{1,1}, w_2 + \frac{1}{2}k_{1,2}, \ldots, w_m + \frac{1}{2}k_{1,m})$.

Step 7. For $j = 1, 2, \ldots m$ set $k_{3,j} = hf_j(t + \frac{h}{2}, w_1 + \frac{1}{2}k_{2,1}, w_2 + \frac{1}{2}k_{2,2}, \ldots, w_m + \frac{1}{2}k_{2,m})$.

Step 8. For $j = 1, 2, \ldots m$ set $k_{4,j} = hf_j(t + h, w_1 + k_{3,1}, w_2 + k_{3,2}, \ldots, w_m + k_{3,m})$.

Step 9. For $j - 1, 2, \ldots m$ set $w_j - w_j + (k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j})/6$.

Step 10. Set $t = a + ih$.

Step 11. OUTPUT $(t, w_1, w_2, \ldots, w_m)$.

Step 12. STOP.

6.2. **Runge-Kutta Explained.** We are trying to solve a system of differential equations with initial values at $(N + 1)$ equally spaced numbers in the interval $[a, b]$. Our system is:

$$u_j' - f_j(t, u_1, u_2, \ldots, u_m),\ a \leq t \leq b, \text{ with } u_j(a) - \alpha_j,$$

for $j - 1, 2, \ldots, m$. We input the endpoints, a and b; the number of equations m; integer N; and initial conditions $\alpha_1, \ldots, \alpha_m$.

Step 1: We create a time step, h, by dividing the range into N pieces. So $h - (b - a)/N$. We will use t as the variable for the time at each point, so t will increase throughout the process. We start t at a, the beginning of the range.

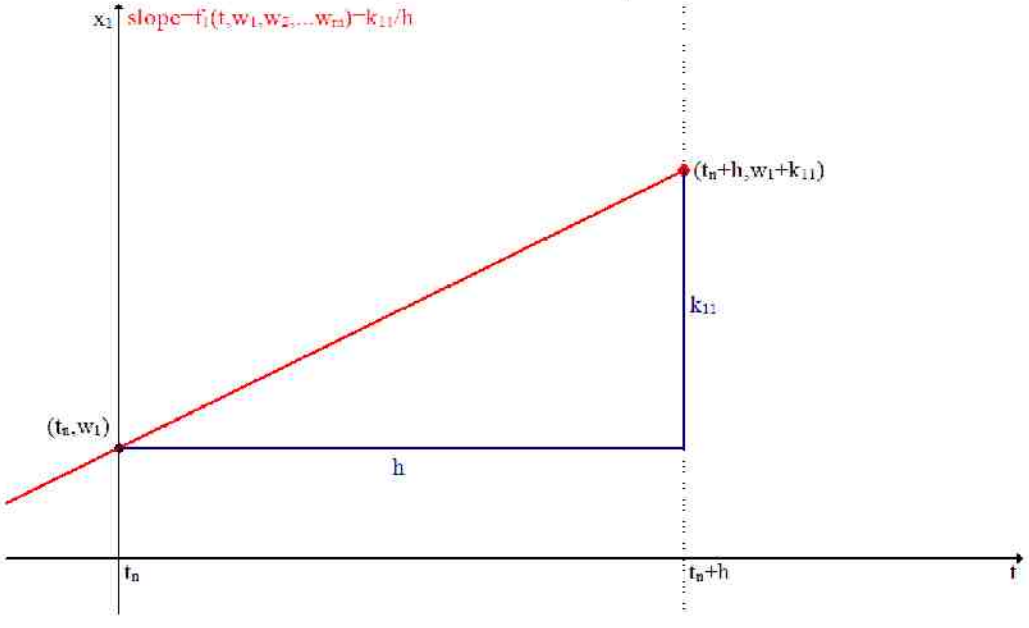
Step 2: We will let each point be of the form $(t, w_1, w_2, \ldots, w_m)$. The first point consists of the initial values we were given, so we set $w_j = \alpha_j$ for $j - 1, 2, \ldots, m$.

Step 3: We output this first point.

Step 4: This is the loop statement. We do Steps 5 through 11 until we have reached the end of our range.

Step 5: We would like to know what the next point is. We can estimate it by using the slope at the current point. Figure 41 depicts this for one of the equations, starting at time t_n. The slope is the derivative, which is our function f_j, so we just evaluate this function given our current point, $(t, w_1, w_2, \ldots, w_m)$. We can introduce another set of variables, $k_{1,j}$, that satisfy:

FIGURE 41. Runge-Kutta Step 5



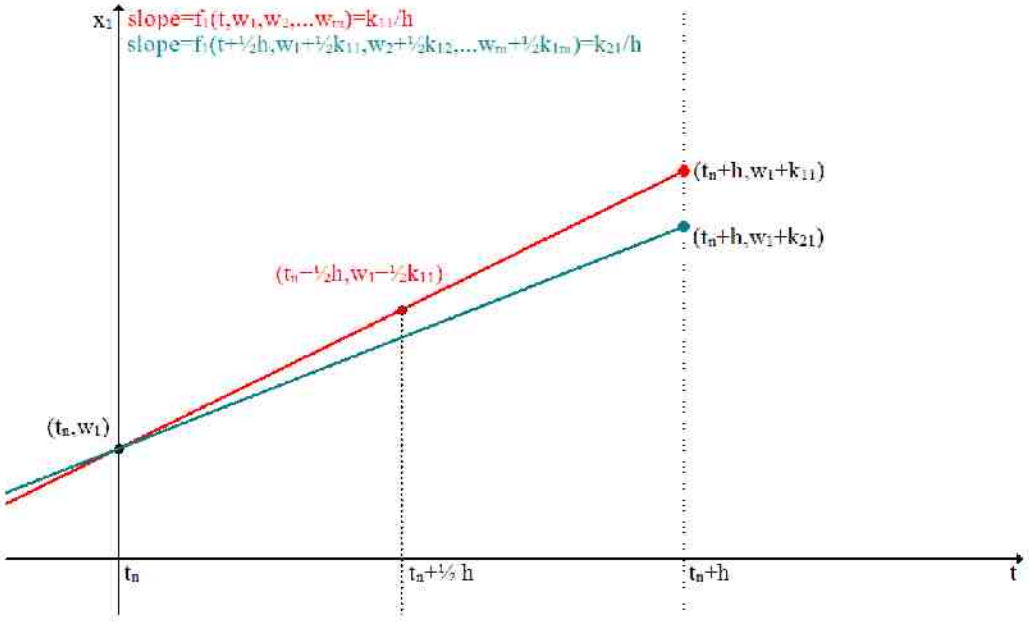
$$slope = f_j(t, w_1, w_2, \ldots, w_m) = \frac{rise}{run} = \frac{k_{1,j}}{h}$$

Then if we rearrange the equation above, we get

$$k_{1,j} = h f_j(t, w_1, w_2, \ldots w_m)$$

In this step, we set the values of $k_{1,j}$ effectively saving the rise.

FIGURE 42. Runge-Kutta Step 6



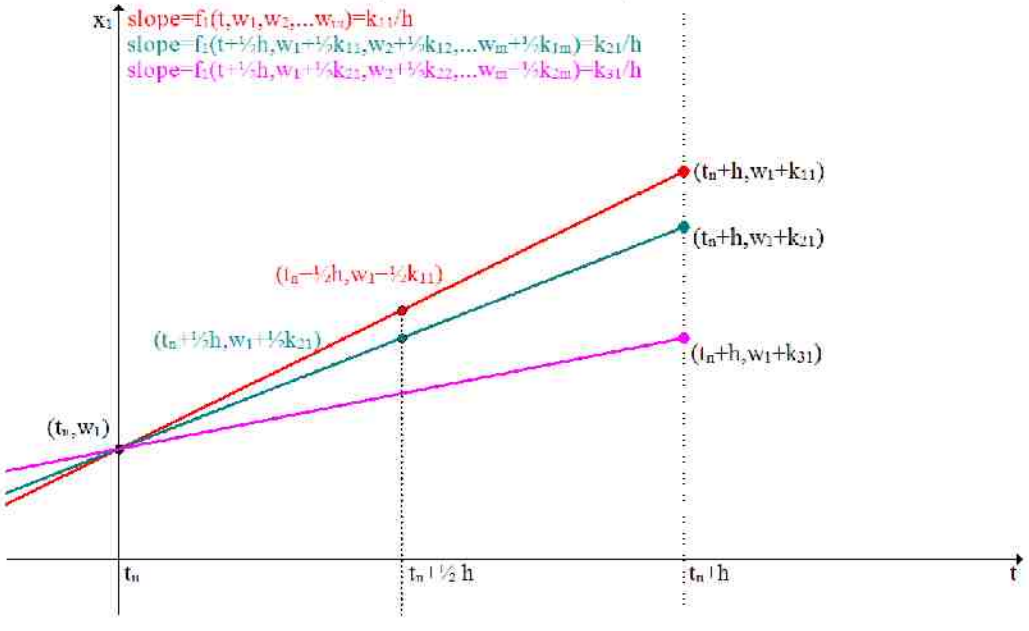
Step 6: This is similar to Step 5. Instead of evaluating the slope at our first point, we take the midpoint between the first point and the predicted point from Step 5, $\left(t + \frac{h}{2}, w_1 + \frac{1}{2}k_{1,1}, w_2 + \frac{1}{2}k_{1,2}, \ldots, w_m + \frac{1}{2}k_{1,m}\right)$. When we evaluate the derivative here, we get the slope of another line, as depicted

in Figure 42. Now we expand our k variable array to $k_{2,j}$, and use the equation below to set its values.

$$slope \quad f_j\left(t+\frac{h}{2}, w_1+\frac{1}{2}k_{1,1}, w_2+\frac{1}{2}k_{1,2}, \quad , w_m+\frac{1}{2}k_{1,m}\right) \quad \frac{rise}{run} \quad \frac{k_{2,j}}{h}$$

$$k_{2,j} = hf_j\left(t \mid \frac{h}{2}, w_1 \mid \frac{1}{2}k_{1,1}, w_2 \mid \frac{1}{2}k_{1,2}, \ldots, w_m \mid \frac{1}{2}k_{1,m}\right)$$

FIGURE 43. Runge-Kutta Step 7

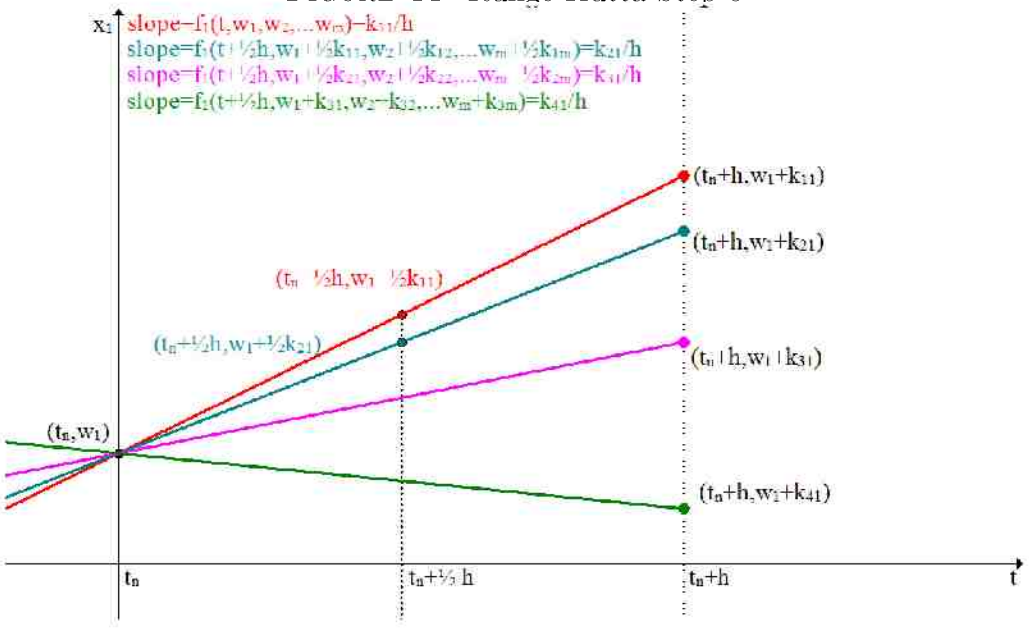


Step 7: The process of Step 5 is repeated again, but this time we use the midpoint on the newest line to evaluate the derivative at. This can be seen graphically in Figure 43.

$$slope \quad f_j\left(t+\frac{h}{2}, w_1+\frac{1}{2}k_{2,1}, w_2+\frac{1}{2}k_{2,2}, \ldots, w_m+\frac{1}{2}k_{,m}\right) \quad \frac{rise}{run} \quad \frac{k_{3,j}}{h}$$

$$k_{3,j} = hf_j\left(t \mid \frac{h}{2}, w_1 \mid \frac{1}{2}k_{2,1}, w_2 \mid \frac{1}{2}k_{2,2}, \ldots, w_m \mid \frac{1}{2}k_{2,m}\right)$$

FIGURE 44. Runge-Kutta Step 8



Step 8: The slope is determined a final time. Step 5 used a point at the beginning of the interval from t to $t + h$, Steps 6 and 7 used points in the middle, and this step uses a point at the end of the interval. The point we use is on the previous line, at the end of the interval, $(t + h, w_1 + k_{3,1}, w_2 + k_{3,2}, \ldots, w_m + k_{3,m})$. When we evaluate the derivative, we get the equations

below.

$$slope \quad f_j(t+h, w_1 + k_{3,1}, w_2 + k_{3,2}, \ldots, w_m + k_{3,m}) \quad \frac{rise}{run} \quad \frac{k_{4,j}}{h}$$

$$k_{4,j} = h f_j(t + h, w_1 + k_{3,1}, w_2 + k_{3,2}, \ldots, w_m + k_{3,m})$$

This step is shown in Figure 44.

Step 9: Now we have four possible points, which means four different rises. We take a weighted average of the rises, $(k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j})/6$. There is more weight placed on the values from the two middle points. We now use this weighted rise as our rise. We take the current point, w_j, and add on $(k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j})/6$, and this becomes the new current point.

$$w_j \quad w_j + (k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j})/6$$

Step 10: We move the current time by adding one time step.

Step 11: We output the current point.

Steps 5 through 11 repeat until all the points are outputted. Then the program can take these points to use for graphing.

6.3. **Modification to the Runge-Kutta Method.** We have to make one modification to the method shown in Sections 6.1 and 6.2. Because our program deals with populations, we cannot have negative values. To account for this, we add a Step 9.5.

Step 9.5. If $w_j \leq 0$, set $w_j \quad 0$ and set coefficients of f_j to 0.

If the value falls negative, we set it to 0. We set the coefficients to 0 so that the derivative is 0 from then on, and no more change will occur.

7. Stochastic Differential Equations (SDE's)

The Logistic Growth Model with Harvesting and Randomness uses stochastic differential equations (SDE's) to incorporate randomness. The SDE's were created by modifying the ordinary differential equations (ODE's) used in the Logistic Growth Model with Harvesting. Here we will discuss how we can modify an ODE to get an SDE, and then how we approximate solutions to that SDE. We will start by looking at the simple ODE for the Continuous Exponential Model, and then extend our results to the more complex ODE's used for the harvesting model.

7.1. **Creating the SDE.** Let us consider the ODE used for the Continuous Exponential Model.

$$\frac{dp}{dt} = rp$$

OR

$$\frac{dp}{dt} = bp - dp$$

Here p is the population size, b is the birth rate, d is the death rate, and $r = b - d$ is the growth rate. We commonly see the first equation above, but the second equation is preferable right now for reasons we will see shortly. We want to modify this ODE by adding on a random piece. Letting $W(t)$ represent the Wiener Process, discussed in Section 7.2, the SDE associated with this ODE is

$$dp = (b - d)p\,dt + \sqrt{bp + dp}\,dW(t) \tag{5}$$

E. J. Allen showed how to derive this equation[2]; we need to understand it so we can create SDE's for other ODE's.

We can start towards an SDE by taking our ODE and adding on random variables. We add one random variable, X_1, to account for fluctuations in the births, and another, X_2, for the deaths.

$$dp = (b - d)p\,dt + X_1(t) + X_2(t)$$

The birth process in the natural world can be considered a Poisson process, as can the death process[1]. The mean and variance are equal in a Poisson process. The mean of the births is bp, so the variance of X_1 is bp. The mean of the deaths is dp, so the variance of X_2 is dp. Since we are considering births and deaths to be independent, the variance combined is the sum of the variances. We can create a new variable, X_3 with mean 0 and variance $bp + dp$ which captures the entire variance of population change.

$$dp = (b - d)p\,dt + X_3(t)$$

Using $dW(t)$ will give us random normal variables with mean 0 and variance 1. If we multiply this by the square root of the variance we desire, we effectively get random normal variables with mean 0 and the correct variance.

$$dp = (b - d)p\,dt + \sqrt{bp + dp}\,dW(t) \tag{5}$$

This is the SDE for the exponential model. Note that the variance is the sum of the absolute-value of the terms of the ODE. Now we want to approximate solutions for it, but to do that we first have to look at the Wiener Process.

7.2. **Wiener Process.** The Wiener process $W(t)$ has a few properties that will be useful in solving the SDE. The value of the Wiener Process at any time t is a random variable with a normal distribution that has mean 0 and variance t.

$$W(t) \sim \mathcal{N}(0, t)$$

The difference between the values at two times, t and s with $t > s$, is a random variable with a normal distribution that has mean 0 and variance $t - s$.

$$W(t) - W(s) \sim \mathcal{N}(0, t - s)$$

We can use this knowledge to find the difference in values between time t and time $t + \Delta t$.

$$W(t + \Delta t) - W(t) \sim \mathcal{N}(0, \Delta t)$$

If we want a normal distribution with mean 0 and variance 1 instead, we can create a new random variable η_i with that distribution.

$$\eta_i \sim \mathcal{N}(0, 1)$$

$$W(t + \Delta t) - W(t) - \eta_i \sqrt{\Delta t} \tag{6}$$

7.3. **Approximating Solutions of the SDE.** We now take our SDE, Equation 5, and approximate solutions for it. We start by changing the derivative into discrete steps.

$$dp \quad (b - d)p\ dt + \sqrt{bp + dp}\ dW(t) \tag{5}$$

$$\Delta p \quad (b - d)p\ \Delta t + \sqrt{bp + dp}\ \Delta W(t)$$

We will start with one point, $p(i)$, and move a small step, Δt, to get the new point, $p(i + \Delta t)$. We do this by adding Δp to $p(i)$.

$$p(i + \Delta t) \quad p(i) + \Delta p$$

$$p(i + \Delta t) = p(i) + (b - d)p\ \Delta t + \sqrt{bp + dp}\ W(i + \Delta t) - W(i)|$$

We can combine this with Equation 6 to get:

$$p(i + \Delta t) \quad p(i) + (b - d)p\ \Delta t + \sqrt{bp + dp}\sqrt{\Delta t}\ \eta_i$$

We could use this equation to obtain an approximate solution to the SDE, by generating random numbers η_i.

7.4. **SDE's in the program.** The previous sections used the Continuous Exponential Model ODE to create an SDE and approximate a solution to it. The equations we actually used in the program that needed the stochasticity were the equations from the Logistic Growth Model with Harvesting. There were four equations used:

$$\frac{dp}{dt} - rp\left(1 - \frac{p}{K}\right) - H \text{ (Constant Harvesting without Threshold)}$$

$$\frac{dp}{dt} = rp\left(1 - \frac{p}{K}\right) - hp \text{ (Proportional Harvesting without Threshold)}$$

$$\frac{dp}{dt} - rp\left(1 - \frac{p}{K}\right)\left(\frac{p}{T} - 1\right) - H \text{ (Constant Harvesting with Threshold)}$$

$$\frac{dp}{dt} - rp\left(1 - \frac{p}{K}\right)\left(\frac{p}{T} - 1\right) - hp \text{ (Proportional Harvesting with Threshold)}$$

We will only look at one of these equations, the one for constant harvesting with a threshold. To create an SDE from this equation, we start by expanding out the factors.

$$\frac{dp}{dt} - -rp + \frac{rp^2}{K} + \frac{rp^2}{T} - \frac{rp^3}{KT} - H$$

We decided to make the assumption that in the same way as the exponential model, each term is susceptible to some variance, and we want random numbers with a variance that is the sum of the absolute-values of the terms. The program allows the user to incorporate randomness into

the growth or into the harvesting, or both. We need to use two separate Wiener Processes.

$$dp \quad \left[-rp + \frac{rp^2}{K} + \frac{rp^2}{T} - \frac{rp^3}{KT} - H\right] dt$$

$$+ \sqrt{\left(rp + \frac{rp^2}{K} + \frac{rp^2}{T} + \frac{rp^3}{KT}\right)} dW_1(t) + \sqrt{H}\, dW_2(t)$$

This is our SDE. To approximate solutions, we use the process in Section 7.3. The two Wiener processes will each produce random variables. We have one associated with the growth, η_i, and one associated with the harvesting, μ_i.

$$p(i + \Delta t) = p(i) + \left[-rp(i) + \frac{rp(i)^2}{K} + \frac{rp(i)^2}{T} - \frac{rp(i)^3}{KT} - H\right] \Delta t$$

$$+\eta_i \sqrt{\left(rp(i) + \frac{rp(i)^2}{K} + \frac{rp(i)^2}{T} + \frac{rp(i)^3}{KT}\right) \Delta t} + \mu_i \sqrt{H \Delta t}$$

There is one more thing we need to incorporate into the model. In the program, we allow the user to select "High," "Medium," or "Low" randomness. We use a generalized Wiener Process to do this by including scalars, s_1 and s_2, which take on a value of 1, 5, or 10 based on the user's selection.

$$p(i + \Delta t) = p(i) + \left[-rp(i) + \frac{rp(i)^2}{K} + \frac{rp(i)^2}{T} - \frac{rp(i)^3}{KT} - H\right] \Delta t$$

$$+s_1\eta_i\sqrt{\left(rp(i)+\frac{rp(i)^2}{K}+\frac{rp(i)^2}{T}+\frac{rp(i)^3}{KT}\right)\Delta t}+s_2\mu_i\sqrt{II\Delta t}$$

This is the equation we use to approximate solutions. There are similar equations for the other three variations of this model.

8. Future Extensions

The program and the website are being given to the CSUCI Math and ESRM Departments. These departments will maintain and improve the project as they desire. Here are some possible ideas for future improvement and expansion:

- Add more models, including models dealing with metapopulations.
- Create more relevant examples, or examples using actual data. Some of the current examples are not very realistic.
- Change the layout of the program and the website.
- Change the automatic scaling for the exponential models. The user may not realize a change has occurred because the scale automatically adjusts. Alternately, give the user a notice whenever the scale changes.
- Put descriptive names on the "Example" buttons.
- Change the names in the Competing Species and Predator-Prey models, according to the example selected.

- Turn some of the example buttons into radio buttons instead.
- Change the photo on each page to one which is associated with one of the examples on that page.

Additionally, the program could be made to:

- Display information when the user hovers over a text box. For example, when the user hovers over the "Threshold" box, it could offer information such as, "The threshold is the limit below which the population will die out. It must be less than the carrying capacity and nonnegative."
- Have editable labels for population names. Then the user would be able to change the names to whatever animals they would like to model.
- Display numbers in a more appropriate format. Currently, if the user inputs "10" for "years", the text box will display "10.0".
- Extend the graphed line to the edge of the graph. Right now it stops short.
- Give some indication that the page is loading. There is sometimes a delay in the loading process, and the user doesn't know that anything is happening.

- Output useful information in case of a crash, and ask the user to email it. This would be helpful if there are any bugs in the program that generally go unobserved.

Appendix A. Program Code: SinglePopulation.java

```
/*
 * SinglePopulation.java
 * Parameter: Model; Possible values: Exponential,
 * Logistic, Threshold, Harvesting, Discrete,
 * Stochastic
 */
package PopDyn;

import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
import javax.swing.ButtonGroup;
import javax.swing.JRadioButton;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class SinglePopulation extends Applet implements
    ActionListener, KeyListener, MouseListener, FocusListener {
    double initPop;
    double growthRate;
    double capacity;
    double threshold;
    double harvestProportion;
    double harvestAmount;
    double years;
    TextField initPopText;
    TextField growthRateText;
    TextField capacityText;
    TextField thresholdText;
    TextField harvestProportionText;
    TextField harvestAmountText;
    TextField yearsText;
    Label errorLabel;//Displays error messages to user
    Button example1Button;
    Button example2Button;
    Button example3Button;
    Button runButton;
    JRadioButton chooseProportionRadio;
    JRadioButton chooseAmountRadio;
    ButtonGroup harvestChoiceButtonGroup;
    JRadioButton growthRandomHighRadio;
    JRadioButton growthRandomMediumRadio;
    JRadioButton growthRandomLowRadio;
    ButtonGroup growthRandomButtonGroup;
```

```
JRadioButton harvestRandomHighRadio;
JRadioButton harvestRandomMediumRadio;
JRadioButton harvestRandomLowRadio;
ButtonGroup harvestRandomButtonGroup;
Checkbox growthCheckbox;
Checkbox harvestCheckbox;
ChartPanel chartPanel;
int model;//number of the model being used, 1-6, assigned
    in init()
int exampleNumber;

@Override
public void init()
//Gets parameter from web and sets model number. Calls
    initialDraw().
{
    String modelName; //Parameter passed in from web
    //If web passes bad parameter, set model to "Logisitc"
    try {
        modelName = getParameter("model");
    } catch (NullPointerException e) {
        modelName = "Logistic";
    }
    if (modelName.equalsIgnoreCase("Discrete")) {
        model = 1;
    } else if (modelName.equalsIgnoreCase("Exponential")) {
        model = 2;
    } else if (modelName.equalsIgnoreCase("Logistic")) {
        model = 3;
    } else if (modelName.equalsIgnoreCase("Threshold")) {
        model = 4;
    } else if (modelName.equalsIgnoreCase("Harvesting")) {
        model = 5;
    } else if (modelName.equalsIgnoreCase("Stochastic")) {
        model = 6;
    } else {
        model = 3;
    }
    initialDraw();
}

private void initialDraw()
/*
    Draws everything based on the model number. Calls
        setVariables(), setTextFields(), createDataset(),
        and createChart().
*/ {
    Label initPopLabel = new Label("Initial Population Size
        (Po):");
    Label growthRateLabel = new Label("Growth Rate (r):");
    Label capacityLabel = new Label("Carrying Capacity (K):
        ");
    Label thresholdLabel = new Label("Threshold (T):");
```

```
Label harvestProportionLabel = new Label("Harvesting
    rate (h):");
Label harvestAmountLabel = new Label("Number harvested
    yearly (H):");
Label randomLabel = new Label("Randomness:");
Label randomLevelLabel = new Label("Amount of
    Randomness:");
Label yearsLabel = new Label("Years (t):");
errorLabel = new Label("");
errorLabel.setForeground(Color.red);
initPopText = new TextField(8);
initPopText.addActionListener(this);
initPopText.addFocusListener(this);
growthRateText = new TextField(8);
growthRateText.addActionListener(this);
growthRateText.addFocusListener(this);
capacityText = new TextField(8);
capacityText.addActionListener(this);
capacityText.addFocusListener(this);
thresholdText = new TextField(8);
thresholdText.addActionListener(this);
thresholdText.addFocusListener(this);
harvestProportionText = new TextField(8);
harvestProportionText.addActionListener(this);
harvestProportionText.addMouseListener(this);
harvestProportionText.addFocusListener(this);
harvestProportionText.addKeyListener(this);
harvestAmountText = new TextField(8);
harvestAmountText.addActionListener(this);
harvestAmountText.addMouseListener(this);
harvestAmountText.addFocusListener(this);
harvestAmountText.addKeyListener(this);
yearsText = new TextField(8);
yearsText.addActionListener(this);
yearsText.addFocusListener(this);
runButton = new Button("Run");
runButton.addActionListener(this);
runButton.addKeyListener(this);
example1Button = new Button("Example 1");
example1Button.addActionListener(this);
example1Button.addKeyListener(this);
example2Button = new Button("Example 2");
example2Button.addActionListener(this);
example2Button.addKeyListener(this);
example3Button = new Button("Example 3");
example3Button.addActionListener(this);
example3Button.addKeyListener(this);
chooseProportionRadio = new JRadioButton("", false);
chooseAmountRadio = new JRadioButton("", false);
harvestChoiceButtonGroup = new ButtonGroup();
harvestChoiceButtonGroup.add(chooseProportionRadio);
```

```
harvestChoiceButtonGroup.add(chooseAmountRadio);
growthRandomHighRadio = new JRadioButton("High", false);
growthRandomMediumRadio = new JRadioButton("Med", false)
    ;
growthRandomLowRadio = new JRadioButton("Low", true);
growthRandomButtonGroup = new ButtonGroup();
growthRandomButtonGroup.add(growthRandomHighRadio);
growthRandomButtonGroup.add(growthRandomMediumRadio);
growthRandomButtonGroup.add(growthRandomLowRadio);
harvestRandomHighRadio = new JRadioButton("High", false)
    ;
harvestRandomMediumRadio = new JRadioButton("Med", false
    );
harvestRandomLowRadio = new JRadioButton("Low", true);
harvestRandomButtonGroup = new ButtonGroup();
harvestRandomButtonGroup.add(harvestRandomHighRadio);
harvestRandomButtonGroup.add(harvestRandomMediumRadio);
harvestRandomButtonGroup.add(harvestRandomLowRadio);
growthCheckbox = new Checkbox("Growth");
harvestCheckbox = new Checkbox("Harvesting");
GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints
    ();
setLayout(gridbag);
//Load Example 1
exampleNumber = 1;
setExample();
XYDataset dataset = createDataset();
JFreeChart chart = createChart(dataset);
chartPanel = new ChartPanel(chart);
constraints.fill = GridBagConstraints.BOTH;
constraints.gridwidth = GridBagConstraints.REMAINDER;
gridbag.setConstraints(chartPanel, constraints);
add(chartPanel);
constraints.gridwidth = 1;
constraints.gridy = 2;
gridbag.setConstraints(initPopLabel, constraints);
add(initPopLabel);
//The unusual widths make the later models fit
    correctly
constraints.gridwidth = 3;
gridbag.setConstraints(initPopText, constraints);
add(initPopText);
constraints.gridwidth = 1;
gridbag.setConstraints(growthRateLabel, constraints);
add(growthRateLabel);
constraints.gridwidth = 2;
gridbag.setConstraints(growthRateText, constraints);
add(growthRateText);
constraints.gridwidth = 1;
/*
```

```
     * Each subsequent model builds upon the previous. For
         example, all
     * models after 2 have capacity. The conditions add in
         the correct
     * components.
     */
    //Add capacity if model isn't discrete or exponential
    if (model > 2) {
        constraints.gridwidth = 2;
        gridbag.setConstraints(capacityLabel, constraints);
        add(capacityLabel);
        constraints.gridwidth = 1;
        gridbag.setConstraints(capacityText, constraints);
        add(capacityText);
    }
    constraints.gridy = 3;
    //Add threshold based on model type
    if (model > 3) {
        gridbag.setConstraints(thresholdLabel, constraints)
            ;
        add(thresholdLabel);
        constraints.gridwidth = 3;
        gridbag.setConstraints(thresholdText, constraints);
        add(thresholdText);
        constraints.gridwidth = 1;
    }
    gridbag.setConstraints(yearsLabel, constraints);
    add(yearsLabel);
    constraints.gridwidth = 2;
    gridbag.setConstraints(yearsText, constraints);
    add(yearsText);
    constraints.gridwidth = 1;
    //Add harvesting to harvesting and stochastic models
    if (model > 4) {
        chooseProportionRadio.setSelected(true);
        constraints.gridy = 4;
        gridbag.setConstraints(chooseProportionRadio,
            constraints);
        add(chooseProportionRadio);
        constraints.gridwidth = 3;
        gridbag.setConstraints(harvestProportionLabel,
            constraints);
        add(harvestProportionLabel);
        constraints.gridwidth = 1;
        gridbag.setConstraints(harvestProportionText,
            constraints);
        add(harvestProportionText);
        constraints.gridy = 5;
        constraints.gridwidth = 1;
        gridbag.setConstraints(chooseAmountRadio,
            constraints);
        add(chooseAmountRadio);
```

```
        constraints.gridwidth = 3;
        gridbag.setConstraints(harvestAmountLabel,
            constraints);
        add(harvestAmountLabel);
        constraints.gridwidth = 1;
        gridbag.setConstraints(harvestAmountText,
            constraints);
        add(harvestAmountText);
    }
    //Add randomness for stochastic model
    if (model == 6) {
        constraints.gridy = 6;
        gridbag.setConstraints(randomLabel, constraints);
        add(randomLabel);
        constraints.gridwidth = 3;
        gridbag.setConstraints(growthCheckbox, constraints)
            ;
        add(growthCheckbox);
        constraints.gridwidth = 1;
        constraints.gridx = 5;
        constraints.gridwidth = 3;
        gridbag.setConstraints(harvestCheckbox, constraints
            );
        add(harvestCheckbox);
        constraints.gridwidth = 1;
        constraints.gridx = GridBagConstraints.RELATIVE;
        constraints.gridy = 7;
        gridbag.setConstraints(randomLevelLabel,
            constraints);
        add(randomLevelLabel);
        gridbag.setConstraints(growthRandomLowRadio,
            constraints);
        add(growthRandomLowRadio);
        gridbag.setConstraints(growthRandomMediumRadio,
            constraints);
        add(growthRandomMediumRadio);
        gridbag.setConstraints(growthRandomHighRadio,
            constraints);
        add(growthRandomHighRadio);
        constraints.gridx = 5;
        gridbag.setConstraints(harvestRandomLowRadio,
            constraints);
        add(harvestRandomLowRadio);
        constraints.gridx = GridBagConstraints.RELATIVE;
        gridbag.setConstraints(harvestRandomMediumRadio,
            constraints);
        add(harvestRandomMediumRadio);
        gridbag.setConstraints(harvestRandomHighRadio,
            constraints);
        add(harvestRandomHighRadio);
    }
    constraints.gridx = 9;
```

```
        gridbag.setConstraints(runButton, constraints);
        add(runButton);
        constraints.gridy = 8;
        constraints.gridx = 1;
        constraints.gridwidth = 3;
        gridbag.setConstraints(example1Button, constraints);
        add(example1Button);
        constraints.gridwidth = 1;
        constraints.gridx = 4;
        gridbag.setConstraints(example2Button, constraints);
        add(example2Button);
        constraints.gridx = 5;
        gridbag.setConstraints(example3Button, constraints);
        add(example3Button);
        constraints.gridy = 9;
        constraints.gridx = 0;
        constraints.gridwidth = GridBagConstraints.REMAINDER;
        gridbag.setConstraints(errorLabel, constraints);
        add(errorLabel);
        setTextFields();
    }

    void setExample() {
        if (exampleNumber == 1) {
            switch (model) {
                case 1://Discrete
                    setVariables(4, 6, 9, 0, 0, 0, 0);
                    break;
                case 2://Exponential
                    setVariables(4, 6, 2.3, 0, 0, 0, 0);
                    break;
                case 3://Logistic
                    setVariables(8, 6, 2.3, 40000, 0, 0, 0);
                    break;
                case 4://Threshold
                    setVariables(18, 350, 0.2, 50000, 400, 0,
                        0);
                    break;
                case 5://Harvest
                    setVariables(10, 500000, 0.09, 3000000, 0,
                        0.2, 0);
                    chooseProportionRadio.setSelected(true);
                    break;
                case 6://Stochastic
                    setVariables(18, 410, 0.2, 50000, 400, 0,
                        0);
                    harvestCheckbox.setState(false);
                    growthCheckbox.setState(true);
                    growthRandomLowRadio.setSelected(true);
                    break;
            }
        } else if (exampleNumber == 2) {
```

```
        switch (model) {
            case 1://Discrete
                setVariables(18, 410, 0.2, 0, 0, 0, 0);
                break;
            case 2://Exponential
                setVariables(18, 410, 0.2, 0, 0, 0, 0);
                break;
            case 3://Logistic
                setVariables(4, 6, 2.3, 40000, 0, 0, 0);
                break;
            case 4://Threshold
                setVariables(18, 410, 0.2, 50000, 400, 0,
                    0);
                break;
            case 5://Harvest
                setVariables(18, 410, 0.2, 50000, 400,
                    0.01, 0);
                chooseProportionRadio.setSelected(true);
                break;
            case 6://Stochastic
                setVariables(18, 425, 0.2, 50000, 400,
                    0.01, 0);
                harvestCheckbox.setState(true);
                growthCheckbox.setState(true);
                growthRandomLowRadio.setSelected(true);
                harvestRandomLowRadio.setSelected(true);
                break;
        }
    } else if (exampleNumber == 3) {
        switch (model) {
            case 1://Discrete
                setVariables(40, 500000, 0.09, 0, 0, 0, 0);
                break;
            case 2://Exponential
                setVariables(40, 500000, 0.09, 0, 0, 0, 0);
                break;
            case 3://Logistic
                setVariables(40, 8000, 0.05, 5000, 0, 0, 0)
                    ;
                break;
            case 4://Threshold
                setVariables(40, 8000, 0.05, 5000, 300, 0,
                    0);
                break;
            case 5://Harvest
                setVariables(10, 500000, 0.09, 3000000, 0,
                    0, 35000);
                chooseAmountRadio.setSelected(true);
                break;
            case 6://Stochastic
                setVariables(40, 1000, 0.05, 5000, 300,
                    0.03, 0);
```

```
                    harvestCheckbox.setState(true);
                    growthCheckbox.setState(true);
                    growthRandomHighRadio.setSelected(true);
                    harvestRandomHighRadio.setSelected(true);
                    break;
                }
            }
        }

    private void setVariables(double y, double i, double r,
        double c, double t,
            double hp, double ha) {
        years = y;
        initPop = i;
        growthRate = r;
        capacity = c;
        threshold = t;
        harvestProportion = hp;
        harvestAmount = ha;
    }

    private void setTextFields() {
        initPopText.setText(Double.toString(initPop));
        growthRateText.setText(Double.toString(growthRate));
        capacityText.setText(Double.toString(capacity));
        thresholdText.setText(Double.toString(threshold));
        harvestProportionText.setText(Double.toString(
            harvestProportion));
        harvestAmountText.setText(Double.toString(harvestAmount
            ));
        yearsText.setText(Double.toString(years));
    }


    private XYDataset createDataset() {
        /*
        For "discrete" and "stochastic", creates data points
            using discrete techniques. For other models, sets
            the coefficients of the differential equations and
            creates a RungeKutta object which holds the data
            points after creation. For this to work, the
            threshold has to be 0 in the exponential model and
            logistic models. harvestAmount and harvestProportion
             have to be 0 unless they are being used. Only one
            of these at a time should be non-zero.
         */
        if (years < 0) {
            years = 1;
        }
        XYSeries series1 = new XYSeries("Population");
        XYSeriesCollection dataset = new XYSeriesCollection();
        if (model == 1)//Discrete
        {
```

```
        double currentPop = initPop;
        series1.add(0, initPop);
        for (int i = 1; i <= years; i++) {
            currentPop += currentPop * growthRate;
            series1.add(i, currentPop);
        }
    } else if (model == 6)//Stochastic
    {
        double growthRandomScalar;
        double harvestRandomScalar;
        int n = 10000;//number of points to generate
        double timeStep = years / n;
        double currentPop = initPop;
        series1.add(0, initPop);
        Random r = new Random();
        double random1 = r.nextGaussian();
        double random2 = r.nextGaussian();
        //Set scalar based on radio selection
        if (growthRandomHighRadio.isSelected()) {
            growthRandomScalar = 10;
        } else if (growthRandomMediumRadio.isSelected()) {
            growthRandomScalar = 5;
        } else {
            growthRandomScalar = 1;
        }
        //Set scalar based on radio selection
        if (harvestRandomHighRadio.isSelected()) {
            harvestRandomScalar = 10;
        } else if (harvestRandomMediumRadio.isSelected()) {
            harvestRandomScalar = 5;
        } else {
            harvestRandomScalar = 1;
        }
        for (int i = 1; i <= n; i++) {
            if (threshold == 0) {
                currentPop += timeStep * (currentPop *
                    growthRate * (1 - currentPop / capacity)
                     - currentPop * harvestProportion -
                    harvestAmount);
                if (growthCheckbox.getState() == true) {
                    currentPop += Math.sqrt((currentPop *
                        growthRate + currentPop * growthRate
                         * currentPop / capacity) * timeStep
                        ) * random1 * growthRandomScalar;
                }
                if (harvestCheckbox.getState() == true) {
                    currentPop += Math.sqrt((currentPop *
                        harvestProportion + harvestAmount) *
                         timeStep) * random2 *
                        harvestRandomScalar;
                }
            } else//if threshold>0
```

```
            {
                currentPop += timeStep * (currentPop *
                    growthRate * (1 - currentPop / capacity)
                     * (currentPop / threshold - 1) -
                    currentPop * harvestProportion -
                    harvestAmount);
                if (growthCheckbox.getState() == true) {
                    currentPop += Math.sqrt((currentPop *
                        growthRate + Math.pow(currentPop, 2)
                         * (growthRate / capacity) * (
                        currentPop / threshold) + Math.pow(
                        currentPop, 3) * (growthRate / (
                        capacity * threshold))) * timeStep)
                        * random1 * growthRandomScalar;
                }
                if (harvestCheckbox.getState() == true) {
                    currentPop += Math.sqrt((currentPop *
                        harvestProportion + harvestAmount) *
                         timeStep) * random2 *
                        harvestRandomScalar;
                }
            }
            series1.add(i * timeStep, currentPop);
            random1 = r.nextGaussian();
            random2 = r.nextGaussian();
        }
    } else {//If model is other than Discrete or Stochastic
        double[] initVal = {initPop};
        double rangeStart = 0;
        double rangeEnd = years;
        int n = 10000;//number of steps to use in Runge-
            Kutta
        double[] coefficients = {0, 0, 0, 0};
        //The coefs of the equation dp/dt = c0+c1p+c2p^2+
            c3p^3
        //For this to work, the threshold has to be 0 in
            the exponential model and logistic models
        if (model == 2) {
            coefficients[1] = growthRate;
        } else if (threshold == 0) {
            //Set the coef for logistic no threshold type
                models
            coefficients[0] = -harvestAmount;
            coefficients[1] = growthRate -
                harvestProportion;
            coefficients[2] = -growthRate / capacity;
        } else {
            //Set the coef for models with a threshold
            coefficients[0] = -harvestAmount;
            coefficients[1] = -growthRate -
                harvestProportion;
```

```
                coefficients[2] = growthRate / capacity +
                    growthRate / threshold;
                coefficients[3] = -growthRate / (capacity *
                    threshold);
            }
            double[][][] coefficientsLarge = {coefficients};
            RungeKutta rk1;
            rk1 = new RungeKutta(coefficientsLarge, 4, initVal,
                rangeStart, rangeEnd, n, 1, "Single");
            for (int i = 0; i < n + 1; i++) {
                series1.add(rk1.point[i][0], rk1.point[i][1]);
            }
            /*
              If RungeKutta experienced an overlimit or
                  extinction, give the user a message
             */
            if (rk1.overLimit || (rk1.extinct && initPop >
                threshold)) {
                String errorMessage = errorLabel.getText();
                if (rk1.overLimit) {
                    errorMessage = errorMessage.concat("Maximum
                        reached.");
                }
                errorMessage = errorMessage.concat("Try
                    adjusting your parameters.");
                if (model == 4) {
                    errorMessage = errorMessage.concat("The
                        carrying capacity should not be too much
                        larger than the threshold.");
                }
                errorLabel.setText(errorMessage);
            }
        }
        dataset.addSeries(series1);
        return dataset;
    }

    private JFreeChart createChart(XYDataset dataset) {
        JFreeChart chart = ChartFactory.createXYLineChart(
                "", // chart title
                "Time in years (t)", // x axis label
                "Population Size (p)", // y axis label
                dataset, // data
                PlotOrientation.VERTICAL,
                false, // include legend
                true, // tooltips//change?
                false // urls
                );
        //Put points on the graph if it's the discrete model
        if (model == 1) {
            XYPlot plot = (XYPlot) chart.getPlot();
```

```
            XYLineAndShapeRenderer renderer = (
                XYLineAndShapeRenderer) plot.getRenderer();
            renderer.setSeriesShapesFilled(0, true);
            renderer.setSeriesShapesVisible(0, true);
        }
        return chart;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == example1Button) {
            exampleNumber = 1;
            redraw(false);
        } else if (e.getSource() == example2Button) {
            exampleNumber = 2;
            redraw(false);
        } else if (e.getSource() == example3Button) {
            exampleNumber = 3;
            redraw(false);
        } else {
            redraw(true);
        }
    }

    @Override
    //This is needed in case a user tabs to a button and pushes
        "Enter"
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_ENTER) {
            if (e.getSource() == runButton) {
                redraw(true);
            } else if (e.getSource() == example1Button) {
                exampleNumber = 1;
                redraw(false);
            } else if (e.getSource() == example2Button) {
                exampleNumber = 2;
                redraw(false);
            } else if (e.getSource() == example3Button) {
                exampleNumber = 3;
                redraw(false);
            }
        }
    }

    public void redraw(boolean userInput)
    /*
     If user input, does error checking on the text boxes and
        sets the variables. If not user input, calls setExample
        () to set the variables. Then calls setTextFields()
        createDataset() and createChart().
     */ {
        double tempDouble;
        String errorMessage = "";
```

```
String initPopError = " Initial population must be a
    positive number.";
String growthRateError = " Growth rate must be a number
    , usually between 0 and 1.";
String capacityError = " Capacity must be a positive
    number.";
String thresholdError = "  Threshold must be a positive
     number, and less than the capacity.";
String harvestProportionError = "  Harvest proportion
    must be a number between 0 and 1.";
String harvestAmountError = "  Harvest amount must be a
     positive number, and less than the capacity.";
String yearsError = "  Years must be positive number,
    and is rounded to tenths of a year.";
if (userInput) {
     if (model > 2) {
          try {
               tempDouble = Double.parseDouble(
                   capacityText.getText());
               if (tempDouble <= 0) {
                    errorMessage = errorMessage.concat(
                        capacityError);
               } else {
                    capacity = tempDouble;
               }
          } catch (NumberFormatException f) {
               errorMessage = errorMessage.concat(
                   capacityError);
          }
     }
     if (model > 3) {
          try {
               tempDouble = Double.parseDouble(
                   thresholdText.getText());
               if (tempDouble < 0 || tempDouble >=
                   capacity) {
                    errorMessage = errorMessage.concat(
                        thresholdError);
               } else {
                    threshold = tempDouble;
               }
          } catch (NumberFormatException f) {
               errorMessage = errorMessage.concat(
                   thresholdError);
          }
     }
     if (model > 4) {
          if (chooseProportionRadio.isSelected()) {
               try {
                    tempDouble = Double.parseDouble(
                        harvestProportionText.getText());
                    if (tempDouble < 0 || tempDouble > 1) {
```

```
                    errorMessage = errorMessage.concat(
                        harvestProportionError);
                } else {
                    harvestProportion = tempDouble;
                }
            } catch (NumberFormatException f) {
                errorMessage = errorMessage.concat(
                    harvestProportionError);
            }
            harvestAmount = 0;
        } else {
            try {
                tempDouble = Double.parseDouble(
                    harvestAmountText.getText());
                if (tempDouble < 0 || tempDouble >=
                    capacity) {
                    errorMessage = errorMessage.concat(
                        harvestAmountError);
                } else {
                    harvestAmount = tempDouble;
                }
            } catch (NumberFormatException f) {
                errorMessage = errorMessage.concat(
                    harvestAmountError);
            }
            harvestProportion = 0;
        }
    }
    try {
        tempDouble = Double.parseDouble(initPopText.
            getText());
        if (tempDouble <= 0) {
            errorMessage = errorMessage.concat(
                initPopError);
        } else {
            initPop = tempDouble;
        }
    } catch (NumberFormatException f) {
        errorMessage = errorMessage.concat(initPopError
            );
    }
    try {
        tempDouble = Double.parseDouble(growthRateText.
            getText());
        growthRate = tempDouble;
    } catch (NumberFormatException f) {
        errorMessage = errorMessage.concat(
            growthRateError);
    }
    try {
        tempDouble = Double.parseDouble(yearsText.
            getText());
```

```
                if (tempDouble <= 0) {
                    errorMessage = errorMessage.concat(
                        yearsError);
                } else {
                    years = tempDouble;
                }
            } catch (NumberFormatException f) {
                errorMessage = errorMessage.concat(yearsError);
            }
        }//close user input case
        else {
            setExample();
        }
        errorLabel.setText(errorMessage);
        years = Math.round(years * 10.0) / 10.0;//rounds to
            tenths place
        setTextFields();
        XYDataset dataset = createDataset();
        JFreeChart chart = createChart(dataset);
        chartPanel.setChart(chart);
    }

    @Override
    public void mouseClicked(MouseEvent e)
    /*
    This is used when a user clicks into one of the harvesting
        boxes, to automatically select the radio button
     */ {
        if (e.getSource() == harvestProportionText) {
            chooseProportionRadio.setSelected(true);
        } else if (e.getSource() == harvestAmountText) {
            chooseAmountRadio.setSelected(true);
        }
    }

    @Override
    public void keyTyped(KeyEvent e)
    /*
     This is used when a user types in one of the harvesting
         boxes, to automatically select the radio button
     */ {
        if (e.getSource() == harvestProportionText) {
            chooseProportionRadio.setSelected(true);
        } else if (e.getSource() == harvestAmountText) {
            chooseAmountRadio.setSelected(true);
        }
    }

    @Override
    public void focusGained(FocusEvent e)
    /*
```

```
        This selects the text in a text box so the user doesn't
            have to delete the current text. It gets automatically
            deleted when the user types new text.
        */ {
            TextField tempField = (TextField) e.getSource();
            tempField.selectAll();
        }

        //The remaining methods are required to be implemented.
        @Override
        public void keyReleased(KeyEvent e) {
        }

        @Override
        public void mousePressed(MouseEvent e) {
        }

        @Override
        public void mouseReleased(MouseEvent e) {
        }

        @Override
        public void mouseEntered(MouseEvent e) {
        }

        @Override
        public void mouseExited(MouseEvent e) {
        }

        @Override
        public void focusLost(FocusEvent e) {
        }
}
```

Appendix B. Program Code: RungeKutta.java

```
//RungeKutta.java
package PopDyn;

public class RungeKutta {

    int numEqs;
    int numberOfCoefficients;
    double[][][] coef;//coefficients of equations
    double[][][] point;//approximated points
    double[] functionVar;//current values to put in the
        function
    String model;
    double timeStep;
    boolean overLimit;
    boolean extinct;

    public RungeKutta(double[][][] c, int nc, double[]
        initialValue, double rangeStart, double rangeEnd, int N,
         int ne, String m) {//c-coef of eqs
```

```
overLimit = false;
extinct = false;
numberOfCoefficients = nc;
numEqs = ne;
model = m;
if (N < 1) {
    N = 2;
}
coef = new double[numEqs][numberOfCoefficients];
for (int i = 0; i < numEqs; i++) {
    System.arraycopy(c[i], 0, coef[i], 0,
        numberOfCoefficients);
}
point = new double[N + 1][numEqs + 1];//1 extra for t
double[] w = new double[numEqs];
double[][] k = new double[4][numEqs];
functionVar = new double[numEqs];

//Step 1
timeStep = (rangeEnd - rangeStart) / N;
double t = rangeStart;

//Step 2
System.arraycopy(initialValue, 0, w, 0, numEqs);

//Step 3
point[0][0] = t;
System.arraycopy(initialValue, 0, point[0], 1, numEqs);

//Step 4
for (int i = 1; i <= N; i++) {

    //Step 5
    System.arraycopy(w, 0, functionVar, 0, numEqs);
    for (int j = 0; j < numEqs; j++) {
        k[0][j] = timeStep * functionEval(j);
    }

    //Step 6
    for (int j = 0; j < numEqs; j++) {
        functionVar[j] = w[j] + 0.5 * k[0][j];
    }
    for (int j = 0; j < numEqs; j++) {
        k[1][j] = timeStep * functionEval(j);
    }

    //Step 7
    for (int j = 0; j < numEqs; j++) {
        functionVar[j] = w[j] + 0.5 * k[1][j];
    }
    for (int j = 0; j < numEqs; j++) {
        k[2][j] = timeStep * functionEval(j);
    }
```

```
//Step 8
for (int j = 0; j < numEqs; j++) {
    functionVar[j] = w[j] + k[2][j];
}
for (int j = 0; j < numEqs; j++) {
    k[3][j] = timeStep * functionEval(j);
}

//Steps 9-11
t = rangeStart + i * timeStep;
point[i][0] = t;
for (int j = 0; j < numEqs; j++) {
    w[j] = w[j] + (k[0][j] + 2.0 * k[1][j] + 2.0 *
        k[2][j] + k[3][j]) / 6.0;
    point[i][j + 1] = w[j];
}

//Step 9.5 to check for harvesting creating
    negative populations
for (int j = 0; j < numEqs; j++) {
    if (point[i][j + 1] < 0) {
        if (model.startsWith("Ecosystem")) {
            //Set coefficients to 0 to prevent
                further change, except those that
                are under division
            point[i][j + 1] = 0;
            coef[j][0] = 0;
            coef[j][2] = 0;
            coef[j][3] = 0;
        } else {
            point[i][j + 1] = 0;
            for (int p = 0; p <
                numberOfCoefficients; p++) {
                coef[j][p] = 0;//This prevents
                    further change in the value
            }
        }
        extinct = true;
    }
    if (point[i][j + 1] > 1000000000)//limit at 1
        billion
    {
        overLimit = true;
        point[i][j + 1] = 1000000000;
        for (int p = 0; p < numberOfCoefficients; p
            ++) {
            coef[j][p] = 0;//This prevents further
                change in the value
        }
    }
}
```

```
    }
}

private double functionEval(int index) {
    double fValue = 0;
    if (model.equals("Single")) {
        for (int i = 0; i < 4; i++) {
            fValue = fValue + coef[0][i] * Math.pow(
                functionVar[0], i);
        }
    } else if (model.equals("Predator")) {
        fValue = coef[index][0] * functionVar[index] + coef
            [index][1] * functionVar[0] * functionVar[1];
    } else if (model.equals("Competing")) {
        fValue = coef[index][0] * functionVar[index] * (1 -
            (functionVar[index] + coef[index][2] *
            functionVar[1 - index]) / coef[index][1]);
    } else if (model.equals("Ecosystem1") || model.equals(
        "Ecosystem2")) {
        fValue = fValue + coef[index][0];//r
        if (index == 0) // -rx/K
        {
            fValue -= coef[0][0] / coef[0][1] * functionVar
                [index];
        } else // -rx/(af)
        if (functionVar[index - 1] > 0) //To avoid division
            by 0, change population size to 1.
        {
            fValue -= (coef[index][0] / (coef[index][1] *
                functionVar[index - 1])) * functionVar[index
                ];
        } else {
            fValue -= coef[index][0] / coef[index][1] *
                functionVar[index];
        }
        if (index < 3) // -by
        {
            fValue -= coef[index][2] * functionVar[index +
                1];
        }
        fValue = fValue * functionVar[index];//This is the
            times x_index
    } else if (model.equals("Ecosystem3")) {
        fValue = fValue + coef[index][0];//r
        switch (index) {
            case 0:
                fValue -= coef[0][0] / coef[0][1] *
                    functionVar[index];// -rx/K
                fValue -= coef[0][2] * functionVar[1];//-by
                fValue -= coef[0][3] * functionVar[2];//-by
                break;
            case 1:
```

```
                    if (functionVar[index - 1] > 0) //-r/(ax)*(
                        x+sy)
                    //To avoid division by 0, change population
                        size to 1.
                    {
                        fValue -= coef[index][0] / coef[index
                            ][1] / functionVar[index - 1] * (
                            functionVar[index] + coef[index][2]
                            * functionVar[2]);
                    } else {
                        fValue -= coef[index][0] / coef[index
                            ][1] * (functionVar[index] + coef[
                            index][2] * functionVar[2]);
                    }
                    fValue -= coef[index][3] * functionVar[3];
                        //-by
                    break;
                case 2:
                    if (functionVar[index - 2] > 0) //-r/(ax)*(
                        x+sy)
                    //To avoid division by 0, change population
                        size to 1.
                    {
                        fValue -= coef[index][0] / coef[index
                            ][1] / functionVar[index - 2] * (
                            functionVar[index] + coef[index][2]
                            * functionVar[1]);
                    } else {
                        fValue -= coef[index][0] / coef[index
                            ][1] * (functionVar[index] + coef[
                            index][2] * functionVar[1]);
                    }
                    break;
                case 3:
                    if (functionVar[index - 2] > 0) //-(r/a)(x/
                        y)
                    //To avoid division by 0, change population
                        size to 1.
                    {
                        fValue -= coef[index][0] / coef[index
                            ][1] / functionVar[index - 2] *
                            functionVar[index];
                    } else {
                        fValue -= coef[index][0] / coef[index
                            ][1] * functionVar[index];
                    }
                    break;
            }fValue = fValue * functionVar[index];//This is the
                times x_index
        }return fValue;
    }
}
```

References

1. M. R. Agnes and J. W. Brian, *Birth and death process modeling leads to the poisson distribution: A journey worth taking*, Primus : Problems, Resources, and Issues in Mathematics Undergraduate Studies **19** (2009), no. 1, pp. 57 73.
2. E. J. Allen, *Introduction to stochastic differential equations in population biology*, MAA PREP, 2010, http://fricative.math.ttu.edu/past/MAAPREP/2010/pix-papers/EAllen.pdf, [Online presentation; accessed 16-October-2012].
3. A. A. Berryman, *The origins and evolution of predator-prey theory*, Ecology **73** (1992), no. 5, pp. 1530 1535.
4. M. S. Boyce, *Population viability analysis*, Annual Review of Ecology and Systematics **23** (1992), no. 1, pp. 481–497.
5. W. E. Boyce and R. C. DiPrima, *Elementary differential equations and boundary value problems*, Wiley, Hoboken, NJ, 2009.
6. R. L. Burden and J. D. Faires, *Numerical analysis*, Thomson Brooks/Cole, Belmont, CA, 2005.
7. A. M. de Roos, *Modeling population dynamics*, University of Amsterdam, Amsterdam, 2011, `http://staff.science.uva.nl/~aroos/downloads/pdf_readers/syllabus.pdf`, [Online; accessed 2-October-2012].
8. T. M. Donovan and C. Welden, *Spreadsheet exercises in ecology and evolution*, Sinauer Associates, Sunderland, MA, 2002.
9. D. C. Duffy, *Competition for nesting space among peruvian guano birds*, The Auk **100** (1983), no. 3, pp. 680–688.

10. J. A. Estes, M. T. Tinker, T. M. Williams, and D. F. Doak, *Killer Whale Predation on Sea Otters Linking Oceanic and Nearshore Ecosystems*, Science **282** (1998), no. 5388, pp. 473 476.

11. H. I. Freedman, *Deterministic mathematical models in population ecology*, Monographs and textbooks in pure and applied mathematics, M. Dekker, New York, 1980.

12. P. Howard, *Modeling with ODE*, Texas A&M University, College Station, TX, 2005, `www.math.tamu.edu/~efendiev/math647_spring05/model_ode.pdf`, [Online; accessed 13-September-2012].

13. D. R. Hundley, *Introduction to mathematical modeling*, `http://people.whitman.edu/~hundledr/courses/M250F03/M250.html`, 2003, [Online; accessed 2-October-2012].

14. C. Jost, G. Devulder, J. A. Vucetich, R. O. Peterson, and R. Arditi, *The wolves of isle royale display scale-invariant satiation and ratio-dependent predation on moose*, Journal of Animal Ecology **74** (2005), no. 5, pp. 809–816.

15. C. J. Krebs, *Ecology : the experimental analysis of distribution and abundance*, third ed., Harper & Row, New York, 1985.

16. D. T. Krohne, *General ecology*, Wadsworth, Belmont, CA, 1998.

17. MathJax, http://www.mathjax.org, 2011, [Online; accessed 7-December-2012].

18. Object Refinery Limited, *JFreeChart*, http://www.jfree.org/jfreechart, 2012, [Online; accessed 7-December-2012].

19. Oracle Corporation, *Learn about Java technology*, http://www.java.com/en/about/, 2012, [Online; accessed 7-December-2012].

20. J. O. Ramsay, G. Hooker, D. Campbell, J. Cao, and J. O. Ramsay, *Parameter estimation for differential equations: A generalized smoothing approach*, Journal of the Royal Statistical Society, Series B (2007).

21. R. E. Ricklefs, *The economy of nature*, sixth ed., W. H. Freeman, New York, 2008.

22. J. A. Vucetich and R. O. Peterson, *Ecological studies of wolves on isle royale, annual report 2011-12*, `http://www.isleroyalewolf.org/wolfhome/ann_rep.html`, 2012, [Online; accessed 23-December-2012].

23. Wikipedia, *American bison — wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=American_bison&oldid=530318833`, 2012, [Online; accessed 3-November-2012].

24. ———, *Generalized Lotka-Volterra equation — wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Generalized_Lotka%E2%80%93Volterra_equation&oldid=502772828`, 2012, [Online; accessed 11-October-2012].

25. ———, *Logistic function — wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Logistic_function&oldid=524948652`, 2012, [Online; accessed 11-October-2012].

26. ———, *Lotka-Volterra equation — wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Lotka%E2%80%93Volterra_equation&oldid=530159256`, 2012, [Online; accessed 11-October-2012].

27. ———, *Population viability analysis — wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Population_viability_analysis&oldid=489617633`, 2012, [Online; accessed 11-October-2012].

28. _____, *Rabbit — wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Rabbit&oldid=529749891, 2012, [Online; accessed 28-October-2012].

29. _____, *Runge-Kutta methods — wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Runge%E2%80%93Kutta_methods&oldid=530632793, 2012, [Online; accessed 22-November-2012].

30. _____, *Trophic level — wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Trophic_level&oldid=525476849, 2012, [Online; accessed 5-November-2012].

31. _____, *Wiener process — wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Wiener_process&oldid=527589687, 2012, [Online; accessed 11-November-2012].