Amy Yu
UID: 905138432
CEE/MAE M20
October 22, 2019

Homework 3

1. Population of two species under competition

1.1 Introduction

The goal in this problem is to use the given Lotka-Volterra equations to find the population of two imaginary species, X and Y, when they are under competition. The populations over time are plotted and printed as a figure, and the population of each species over a certain time is also printed.

1.2 Model and Methods

The discretized equations using the forward Euler method are used to calculate the population of species X and Y, and they were calculated as follows:

$$\frac{x_{k+1} - x_k}{\Delta t} = 3x_k - 0.3x_k x_k - 2x_k y_k$$
$$\frac{y_{k+1} - y_k}{\Delta t} = -2y_k + 1.5x_k y_k - 0.2y_k y_k$$
$$x_{k+1} = x_k + \Delta t(3x_k - 0.3x_k x_k - 2x_k y_k)$$
$$y_{k+1} = y_k + \Delta t(-2y_k + 1.5x_k y_k - 0.2y_k y_k)$$

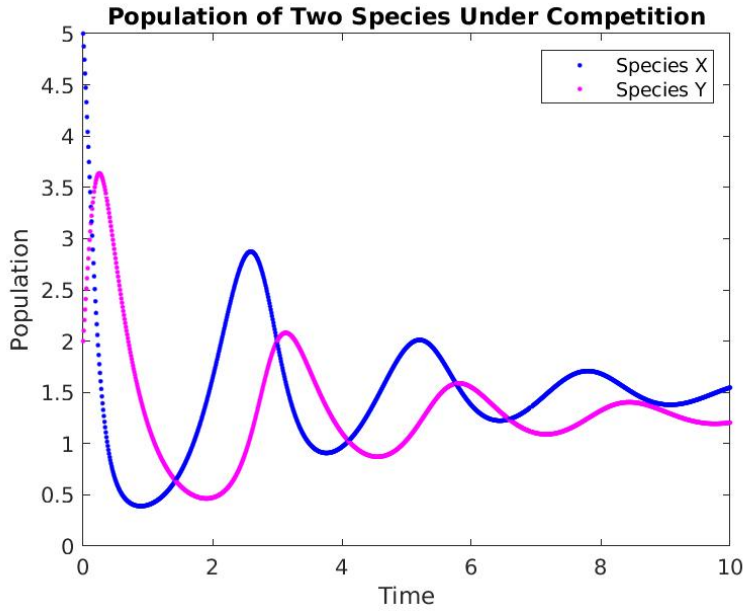
The values of the given coefficients are set as variables in order to prevent them from accidentally being changed. The same was done with the given times, values, and delta T. The number of total steps was calculated and used in a `for` loop. The loop allows us to plot the points, and the values of x and y are updated according to the equations above. The `hold on` function is used to ensure that the previous points remain on the graph as new points are plotted. The curve for each species is set to a different color and a legend is created. The title and axes labels are also created. The `save as` function is used to save the figure as an image, and the final populations of species x and y are printed to the screen.

1.3 Calculations and Results

When the program is executed, the following is printed to the screen:

```
The final populations are:
Species X =    1.54671
Species Y =    1.20538
```

The following figure is printed to the screen:



This shows the graph with a delta t of 0.01. Using the `tic` and `toc` commands, the time is printed to the screen:

```
Elapsed time is 2.646848 seconds.
```

By varying the value of delta t, we can see that the smaller the time difference, the longer the code takes to execute. This makes sense, because the for loops would have to go through more runs and more points would be plotted on the graph.

```
delta_t = 0.0001;
Elapsed time is 215.450264 seconds.

delta_t = 0.001:
Elapsed time is 21.237884 seconds.

delta_t = 0.1:
Elapsed time is 0.858911 seconds.

delta_t = 1:
Elapsed time is 0.692996 seconds.
```

As the value of delta t is increased, less points are plotted and it becomes increasingly harder to see a correlation in the figure.

1.4 Discussion

The figures show that the peaks in the population of Species Y lag slightly behind that of Species X. It can be seen that as time passes, the population of both species begin to converge and have very similar values. The chosen value of delta t is appropriate, as it allows the figure to have enough points plotted and provides a fairly accurate population calculation.

2. Improved computation of population

2.1 Introduction

The goal in this problem is to use an improved formula to calculate the population of two species under competition. A function is given to consider the effect of other factors of the population of Species X. The results are plotted and the figure and final populations are printed to the screen.

2.2 Models and Methods

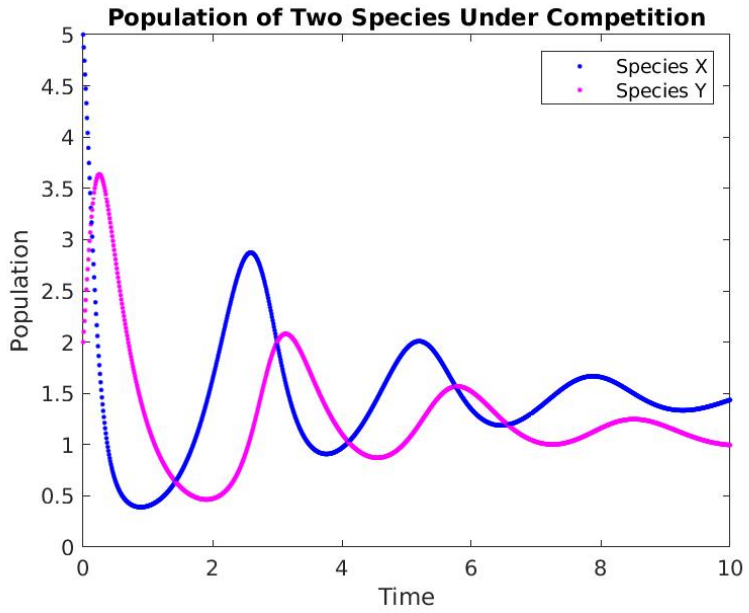
The method in problem one was modified by added another section in the for loop. An if statement was added to make sure that the value of alpha is altered as soon as the beginning time of the effect is reached. Additionally, as new variable is initialized to the time that the new effect begins taking place. This value is updated by delta t after each iteration of the `for` loop, and the same thing is done for the value of `ti`.

2.3 Calculations and Results

When the program is executed, the following is printed to the screen:

```
Species X =    1.43639
Species Y =    0.99600
Elapsed time is 2.803072 seconds.
```

The following figure is printed to the screen:




```
delta_t = 0.0001;
Elapsed time is 222.788275 seconds.
```

```
delta_t = 0.001:
Elapsed time is 27.971918 seconds.

delta_t = 0.1:
Elapsed time is 1.276106 seconds.

delta_t = 1:
Elapsed time is 0.692996 seconds.
```

Similarly to the previous problem, less points are plotted and it becomes increasingly harder to see a correlation in the figure at delta t is increased.

2.4 Discussion

The graphs between this problem and the previous problem are very similar in the beginning, and they begin to slightly differ over time, as soon as the effect kicks in. Towards the end, the population of the two species no longer overlap. It can be seen that the time to run the second problem is slightly longer than the first problem. This is likely due to the additional statements in the `for` loop, which increases the run time as the loop is repeated.

3. Permutation

3.1 Introduction

The goal in this problem is to write a script that takes to integers as inputs and outputs the permutation.

3.2 Models and Methods

The script first prompts the user to enter the values of n and r, which will be used in the calculation of the permutation. The user is prompted to enter they values again if they enter a negative number or decimal. Then, two variables are initialized to one and a `for` loop is used to multiply each value by the integer before it, and this will equal $n!$. Another `for` loop is used to do the same thing, but this time multiplying to $n - r$. Lastly, the result is printed as an integer by dividing the $n!$ by $(n - r)!$.

3.3 Calculations and Results

When the program is run with 10 as the value of n and 6 as the value of r, the following is outputted:

```
Enter value of n:
10
Enter value of r:
6
151200
```

By checking with a calculator, we can see that this is the correct answer.

However, when the program is run with 1000 as the value of n and 0 as the value of r, the following is outputted:

```
Enter value of n:
1000
Enter value of r:
0
NaN
```

This is because the factorial of 1000 is too large to be stored by the computer. The correct result should be 1.

3.4 Discussion

Although this program does calculate permutations correctly, is unable to calculate the permutations of larger integers. One way to improve this code could be to set a condition to cancel out certain numbers before calculating the factorial. For example, when n is equal to 1000 and r is equal to 0, the numerator and denominator should cancel out and the answer should be one. However, this program first attempts to calculate the factorial, so the output is given as `NaN`.