


EA4 Lab 1: The Logistics Equation

10/8/19

Adolphus Adams, Preston Collins, Shane Dolan, and Braden Rocio

Certification Page

We certify that the members of the team named below worked on this lab using only our own ideas, possibly with the help from the TAs and our professors, specifically in the writing of the report, the programming and the analysis required for each task. We did not collaborate with any members of any other team nor did we discuss this lab or ask assistance from anyone outside of our team, the TAs and our professors. If we accessed any websites in working on these labs, these websites are explicitly listed in our report (with the exception of websites used only to get information on specific Matlab programs). We understand that there will be severe consequences if this certification is violated and there is unauthorized collaboration.

1. Shane Dolan -  - 10/8/19

2. Braden Rocio -  - 10/8/19

3. Adolphus Adams  - 10/8/19

4. Preston Collins -  - 10/8/19

Introduction

Lab 1: The Logistics Equation

The rate of change of population can be modeled with a differential equation. The rate of change in the population for any given point in time is dependent on the current population and the environment surrounding the population. The net birth rate of a population is affected by factors such as competition and the death rate. The goal of this lab is to model several different population models by solving differential equations using MATLAB. This lab will explore the differences in solving accuracy between different numerical solvers provided within MATLAB. In addition, it will provide lab members more experience with the differential equation tool set that MATLAB provides.

Task 1: Nondimensionalization

The logistic equation:

$$\frac{d\hat{P}}{d\hat{t}} = \hat{P} (kM - k\hat{P}), \quad \hat{P}(0) = \hat{P}_0$$

From the units for \hat{P} (units of population) and \hat{t} (units of time), the units of k and M are:

$$\begin{aligned} \text{Units of } M &: \text{Units of Population} \\ \text{Units of } k &: \frac{1}{(\text{Units of Time})(\text{Units of Population})} \end{aligned}$$

Define P_{ref} and t_{ref} such that:

$$P = \frac{\hat{P}}{P_{ref}}$$

$$t = \frac{\hat{t}}{t_{ref}}$$

Rearrange the equation to be:

$$\hat{P} = P P_{ref}$$

$$\hat{t} = t t_{ref}$$

Through the process of nondimensionalization, explicitly determine P_{ref} , t_{ref} , and P_0 so that it satisfies the equation $\frac{dP}{dt} = P(1 - P)$, $P(0) = P_0$:

$$\frac{d\hat{P}}{d\hat{t}} = P P_{ref} (kM - kP P_{ref})$$

Using the chain rule, find an equivalency for $\frac{d\hat{P}}{d\hat{t}}$:

$$\frac{d}{d\hat{t}} = \frac{d}{dt} * \frac{dt}{d\hat{t}}$$

$$t = \frac{\hat{t}}{t_{ref}} \Rightarrow \frac{dt}{d\hat{t}} = \frac{1}{t_{ref}}$$

$$\Rightarrow \frac{d}{d\hat{t}} = \frac{1}{t_{ref}} * \frac{d}{dt}$$

$$\Rightarrow \frac{d\hat{P}}{d\hat{t}} = \frac{P_{ref}}{t_{ref}} * \frac{dP}{dt}$$

Substituting this into the original equation:

$$\frac{P_{ref}}{t_{ref}} * \frac{dP}{dt} = P P_{ref}(kM - kP P_{ref})$$

$$\frac{dP}{dt} = P t_{ref}(kM - kP P_{ref}) = t_{ref}kMP - t_{ref}P_{ref}kP^2$$

$$\text{Let } t_{ref}kM = 1 \Rightarrow t_{ref} = \frac{1}{kM}$$

$$\text{Let } t_{ref}P_{ref}k = 1 \Rightarrow P_{ref} = M$$

Substituting this into the equation for \hat{P} :

$$P = \frac{\hat{P}}{P_{ref}} \Rightarrow P_0 = \frac{\hat{P}_0}{P_{ref}} = \frac{\hat{P}_0}{M}$$

So in terms of k , M , and \hat{P}_0 :

$$P_{ref} = M$$

$$t_{ref} = \frac{1}{kM}$$

$$P_0 = \frac{\hat{P}_0}{M}$$

Task 1 Analysis:

Now the variables P and t do not have any units and the reference variables P_{ref} and t_{ref} have units that are related to k and M and were used to eliminate the two parameters in the logistic equation.

Task 2: Computations of the Logistic Equation

Introduction

Using MATLAB's included function *ode45* the logistic equation will be numerically solved. In order to make this process easier, a set of constants will be enumerated in a configuration file called *pop.m* that will process the right side of the differential equation. This program will contain essential constants like solve duration, initial conditions, and solve tolerances. The second program to be created will be called *yprime.m* and will evaluate the left hand side of the logistic equation. This program will contain the options for, and pass off to *ode45*.

Using the following parameters:

- $T_{fin} = 200$
- $P_{init} = 0.95$ and 1.05

The plots for solution versus time are:

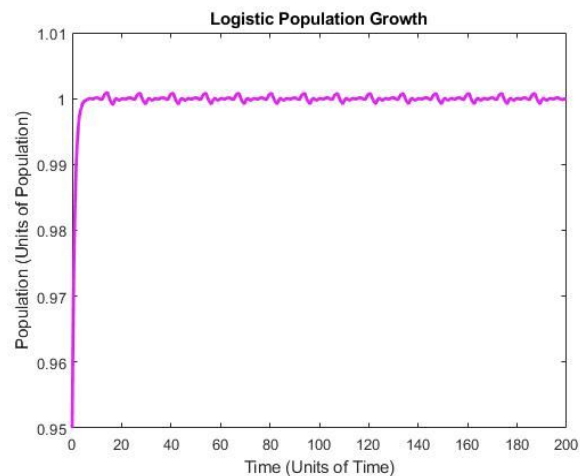


Figure 1: Graph of the solution of the nondimensionalized logistic equation given the initial conditions $t_{fin}=200$, $p_{init}=0.95$ with default *ode45* tolerance

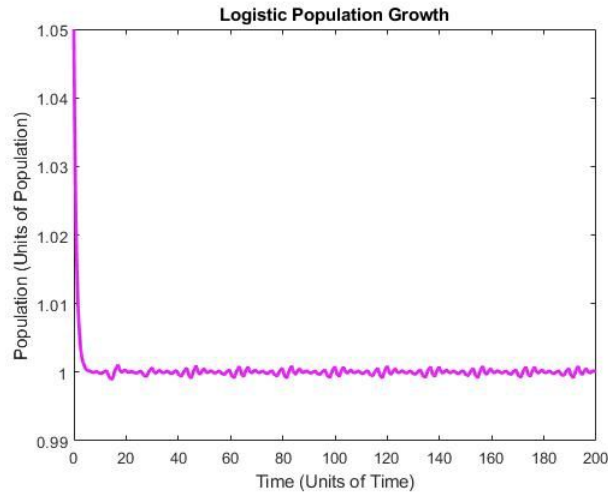


Figure 2: Graph of the solution of the nondimensionalized logistic equation given the initial conditions $t_{fin}=200$, $p_{init}=1.05$ with default ode45 tolerance

Critical Point Analysis

Taking the given function:

$$\frac{dP}{dt} = P(1 - P), \quad P(0) = P_0$$

Call $\frac{dP}{dt} \rightarrow f(P)$:

$$f(P) = P(1 - P)$$

From this, set it equal to 0 and solve for P:

$$0 = P(1 - P)$$

$$0 = P - P^2$$

$$P_c = 1, 0$$

Using $f'(P)$, solve for whether or not the critical points are stable or not:

$$f'(P) = 1 - 2P$$

$$f'(1) = -1 \Rightarrow \text{stable}$$

$$f'(0) = 1 \Rightarrow \text{unstable}$$

This means that the function $P(t)$ stabilizes at $P = 1$.

When looking at figure 1 and 2, the graph does come close to stabilizing at $P = 1$ but has some bumps and oscillations. These are due to the error in ode45. Ode45 is a function to approximate the solution of the differential equation and is not perfect.

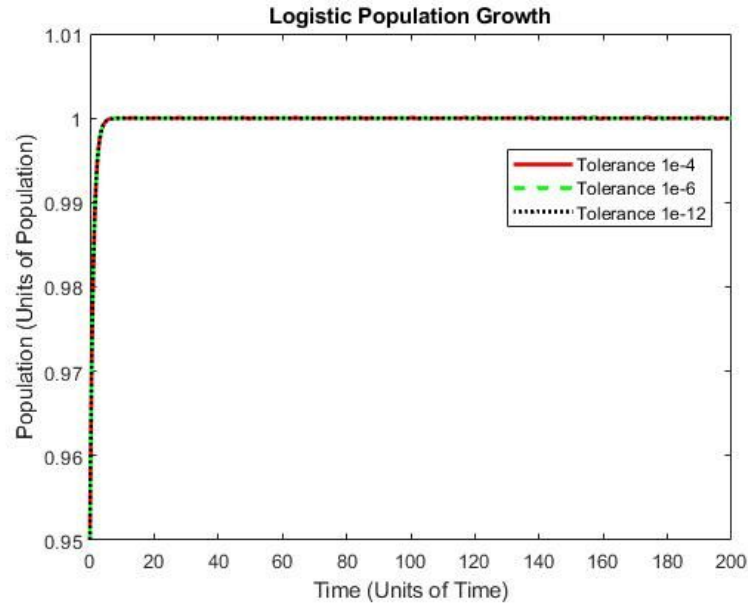


Figure 3: Graph of the solution of the nondimensionalized logistic equation given the initial conditions $t_{fin}=200$, $p_{init}=0.95$ with tolerances $1e-4$, $1e-6$, and $1e-12$

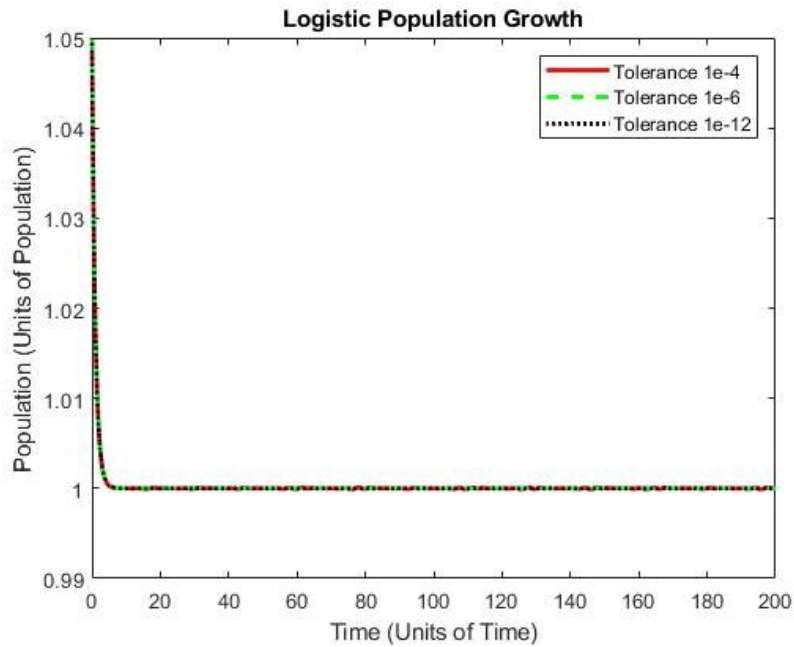


Figure 4: Graph of the solution of the nondimensionalized logistic equation given the initial conditions $t_{fin}=200$, $p_{init}=1.05$ with tolerances $1e-4$, $1e-6$, and $1e-12$

Task 2 Analysis:

One can trust the computations with smaller tolerances because a smaller tolerance reduces the step size in the computations which in turn reduces the error making the computations more accurate. This can be seen in Figures 3 and 4 where the values are much smoother at the stable point $P=1$ than at the same points in Figures 1 and 2, this more accurately represents the logistic equation graph as it approaches but never quite gets to 1. The computations should be trusted because when changing the initial population condition to above and below 1, the graph always normalizes to 1, meaning it is a stable function at the P value 1. One should trust the smaller tolerances over a solution with larger tolerances because the solve was based on more computations. As more computations are performed, the numerical error tends to decrease. The dangers of blindly accepting the results of a computation are that the tolerance selected may not be small enough to provide an accurate solution of the equation that is being modeled. In Figures 1 and 2 the function oscillates at $P=1$ rather than smoothing out as in class and this is confirmed to be inaccurate as when our tolerance gets smaller, the graph becomes smoother at $P=1$. This indicates that not all graphs of solutions are accurate and sometimes need a smaller margin of error (or smaller tolerance) in order to have a graph that better represents the computations.

Task 3: Seasonal Model

Introduction

In task 3, the seasonal dependence of carrying capacity will be examined. It should be noted that the carrying capacity of a population can be dependent on environmental factors. The availability of resources and presence of predators in an ecosystem will certainly affect the maximum population that a certain ecosystem will sustain. In order to model a system in a way similar to reality, a time dependent term is included in the logistic equation. The following model was considered:

$$\frac{dP}{dt} = P \left(1 - \frac{P}{1+bs(t)} \right), P(0) = P_0$$

Where $b \geq 0$ is an input parameter and the function $s(t)$ is defined by:

$$s(t) = \text{mod}(t, \text{dur})$$

This simply bases the carrying capacity off of the presence of an integer remainder within denominator. This makes the carrying capacity of the population modulate up and down which simulates seasonal factors affecting max population. Because this differential equation is now affected by the external input of time, it is a nonautonomous equation.

Multiple different tolerances and parameters were tested when using this model. The following parameters were used for figures 5-8:

$$b = 0.8$$

$$\text{dur} = 0.5$$

$$p_{\text{init}} = 1.13$$

$$t_{\text{fin}} = 25$$

The results were as follows:

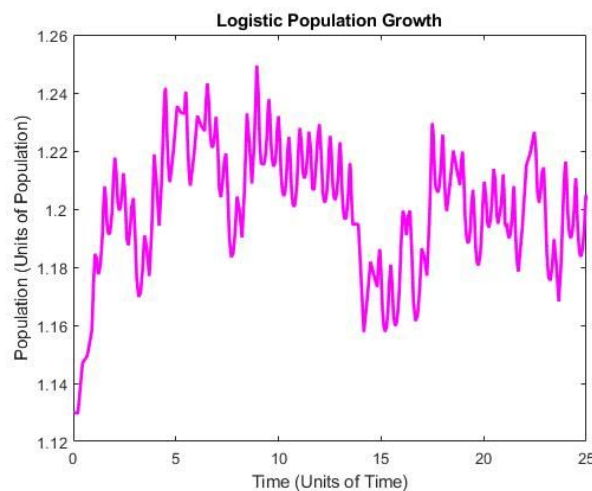


Figure 5: Season model: with default ode45 tolerance and $\text{dur}=.5$

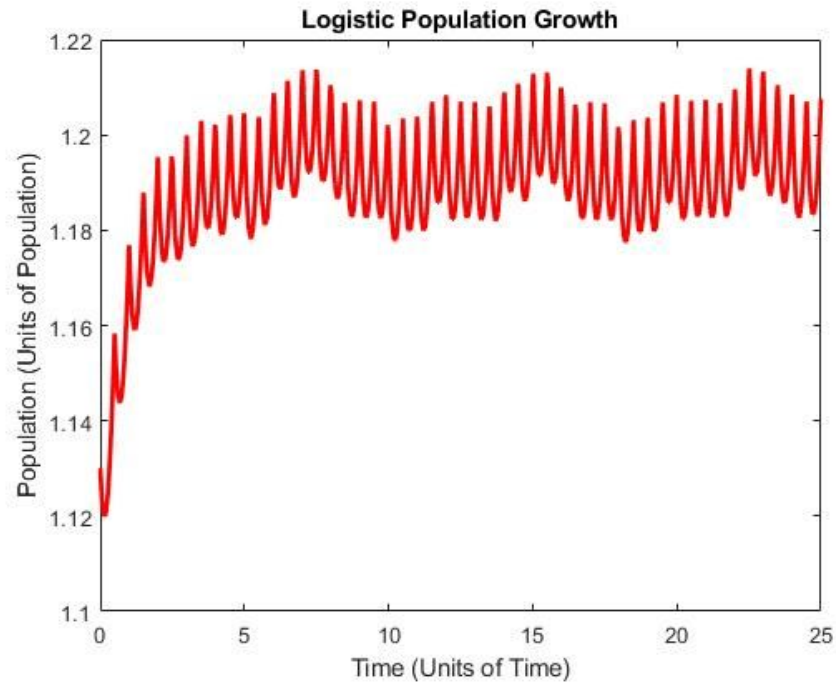


Figure 6: Seasonal model: $tol=1.e-4$ and $dur=.5$

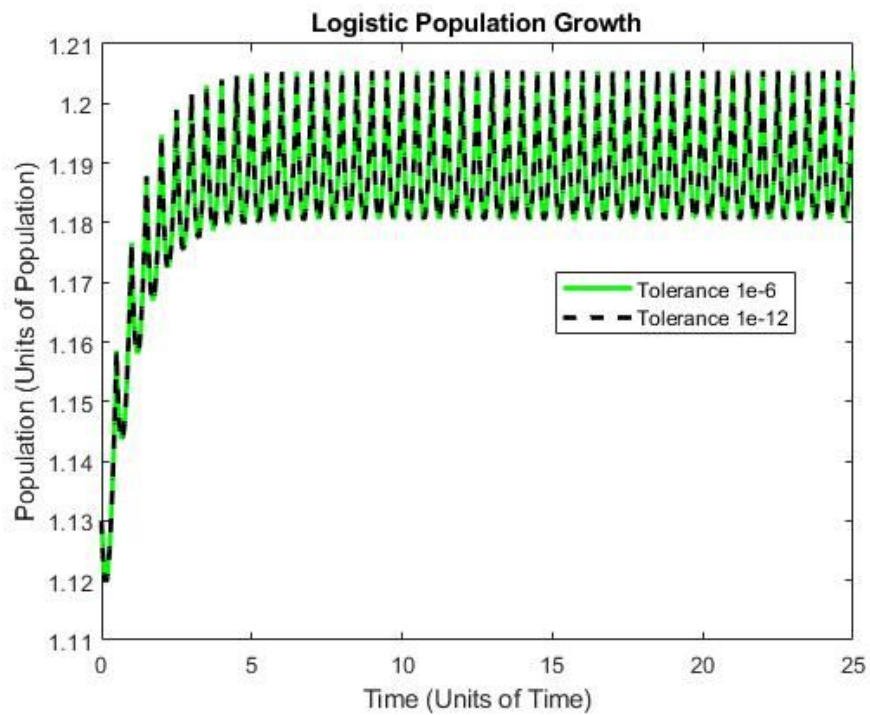


Figure 7: Seasonal model: $tol=1.e-6$ & $tol=1.e-12$ and $dur=.5$

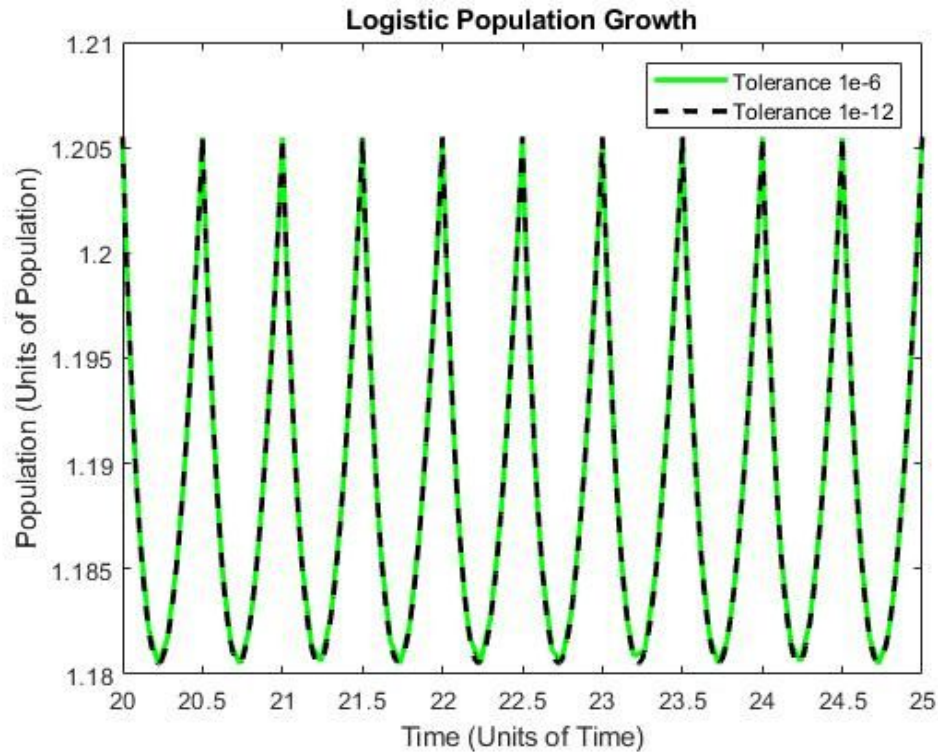


Figure 8: Seasonal model: $tol=1e-6$ & $tol=1e-12$ from $x=20$ to $x=25$ and $dur=.5$

Task 3 Analysis Part 1 (Figures 5-8):

The structure of the solution is one that begins at 1.13 and then oscillates continuously while rapidly increasing until it begins to level out horizontally where the function oscillates between about $P=1.205$ and $P=1.18$. The reason that it does this is because the equation is now a nonautonomous equation meaning there is the independent variable t on the right side of the equation which causes it to explicitly depend on the value t . This causes the sudden jumps (oscillations) that we see in our figures. If one were to believe the first two figures (figures 5 and 6) it would tell us that the population is nearly completely random (in the case of figure 5) or that the oscillations don't have a constant range (between $P=1.18$ and $P=1.205$), as is the case in figure 6. The plots in figure 7 should be believed due to the fact that the tolerance changes from $1e-6$ to $1e-12$ while the graph stays the same meaning that this is as accurate as the graph gets with a very small amount of error. This idea is reinforced when looking at figure 8 which zooms in on $t=20$ to $t=25$ of the graph from figure 7 and shows complete overlap of the two tolerance plots confirming the idea in figure 7 that tolerances $1e-6$ and $1e-12$ both very accurately describe the population growth within the seasonal model.

The following parameters were used for figures 9-12:

$$b = 0.8$$

$$dur = 2$$

$$pinit = 1.13$$

$$tfin = 25$$

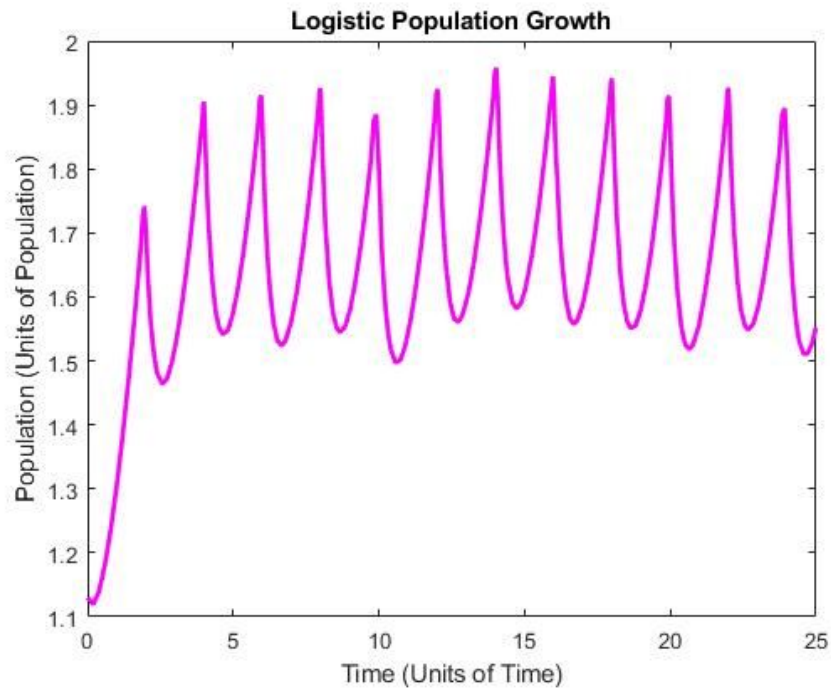


Figure 9: Seasonal model: default tolerance for ode45 and $dur=2$

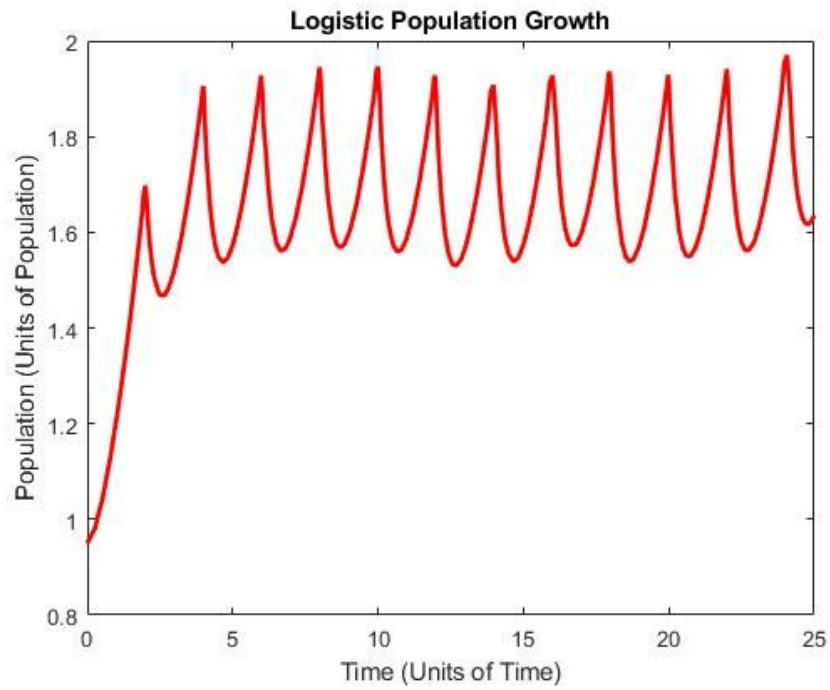


Figure 10: Seasonal model: $tol=1e-4$ and $dur=2$

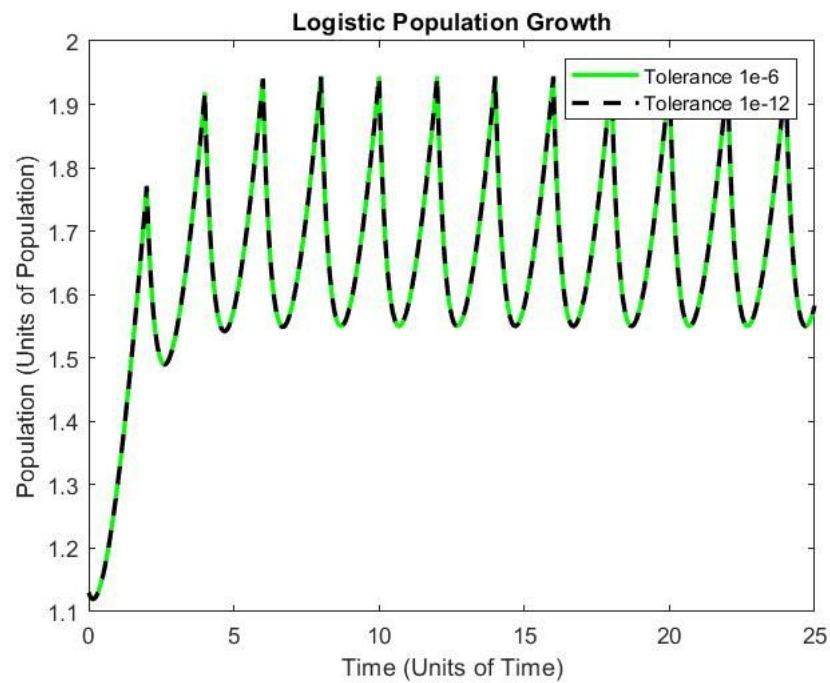


Figure 11: Season model: $tol=1e-6$ and $tol=1e-12$ with $dur=2$

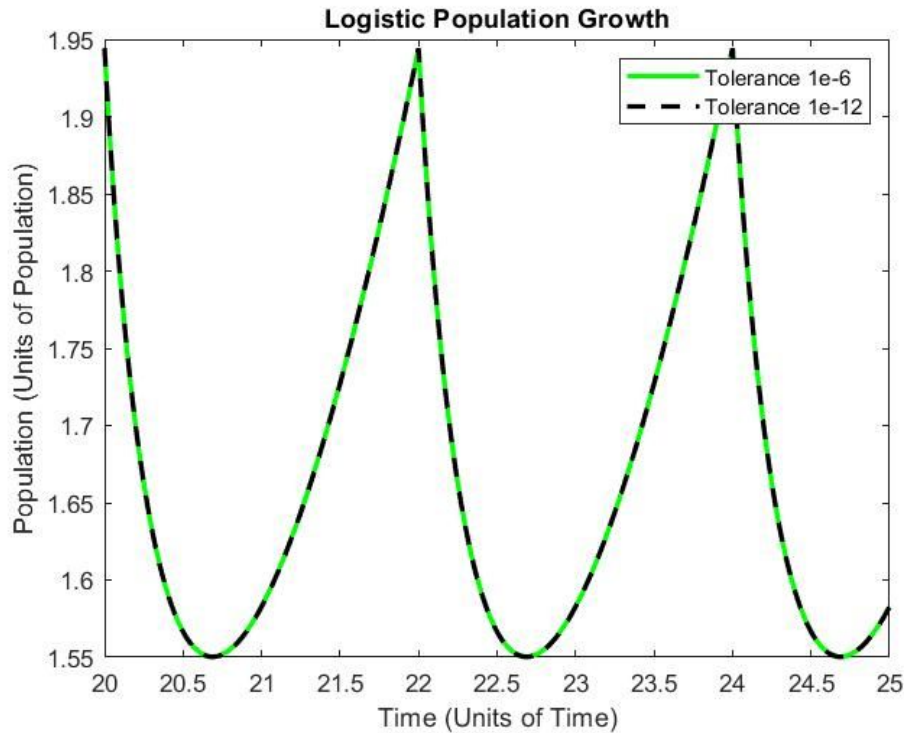


Figure 12: Season model: $tol=1e-6$ and $tol=1e-12$ from $x=20$ to $x=25$ with $dur=2$

Task 3 Analysis Part 2 (Figures 9-12):

When evaluating this equation, it seems that increasing the value of dur increases the base accuracy of the plot as figures 9 and 10 are far more similar to the plots in figure 11 than we saw in the same relationship of figure 5 and 6 to figure 7. This is a reasonable explanation because when the value of dur increases, the time between the carrying capacity jumping is lengthened allowing more time for the solution to stabilize creating a plot that is more accurate with less random jumps. Figures 9 and 10 provide a much more accurate representation of the seasonal model given our initial conditions and global variables, though not quite exact as we can see in figure 11. Once again it is seen that tolerances $1e-6$ and $1e-12$ produce nearly exactly the same plot as seen in figure 12.

The following parameters were used for figures 13-17:

$$b = 0.8$$

$$dur = 0.1$$

$$pinit = 1.03$$

$$tfin = 5$$

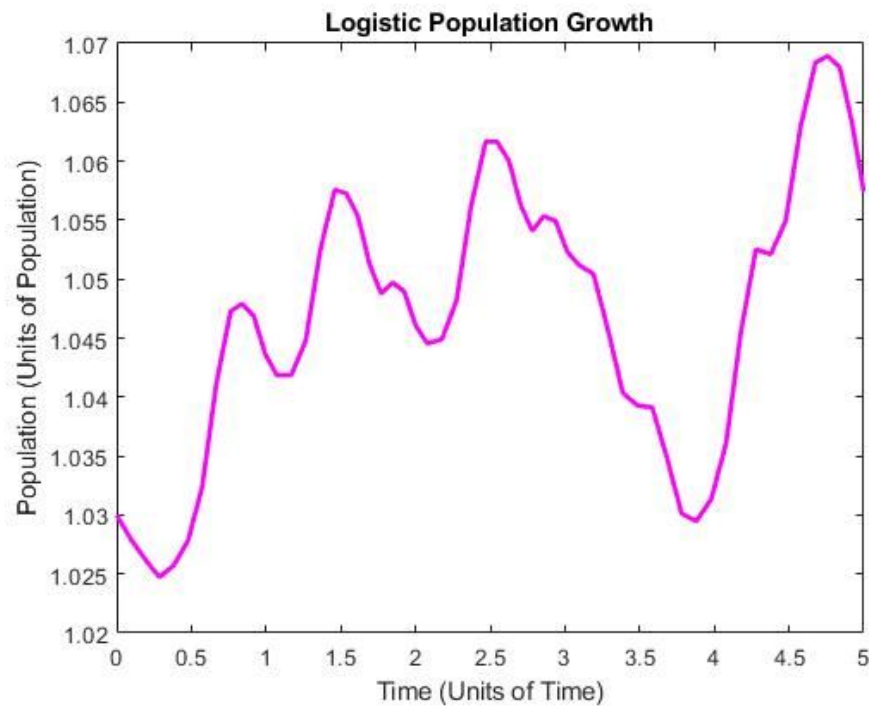


Figure 13: Seasonal model: default ode45 tolerance $pinit$ 1.03 and $dur=.1$ and $tfin=5$

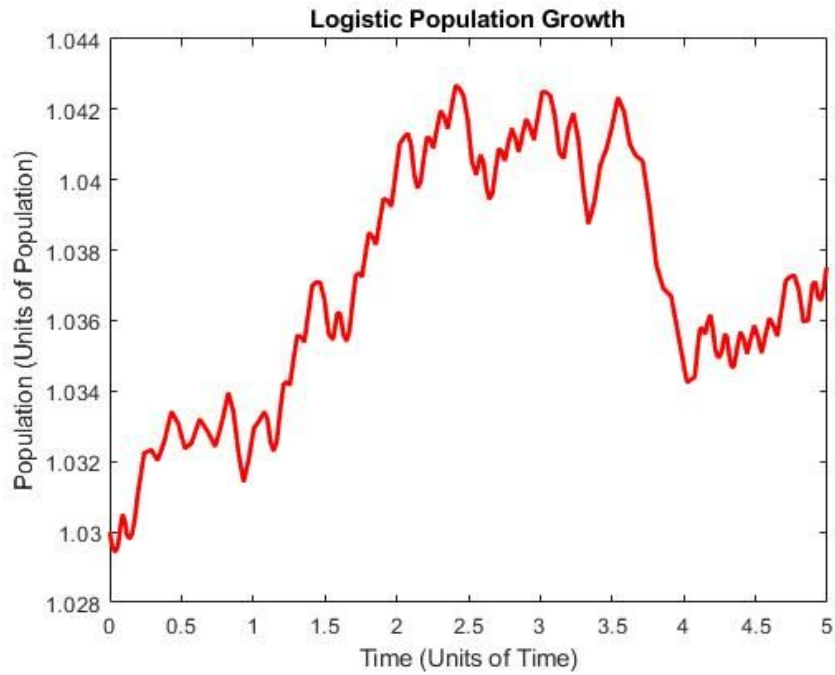


Figure 14: Seasonal model: $tol=1e-4$, $pinit=1.03$, $dur=.1$, and $tfin=5$

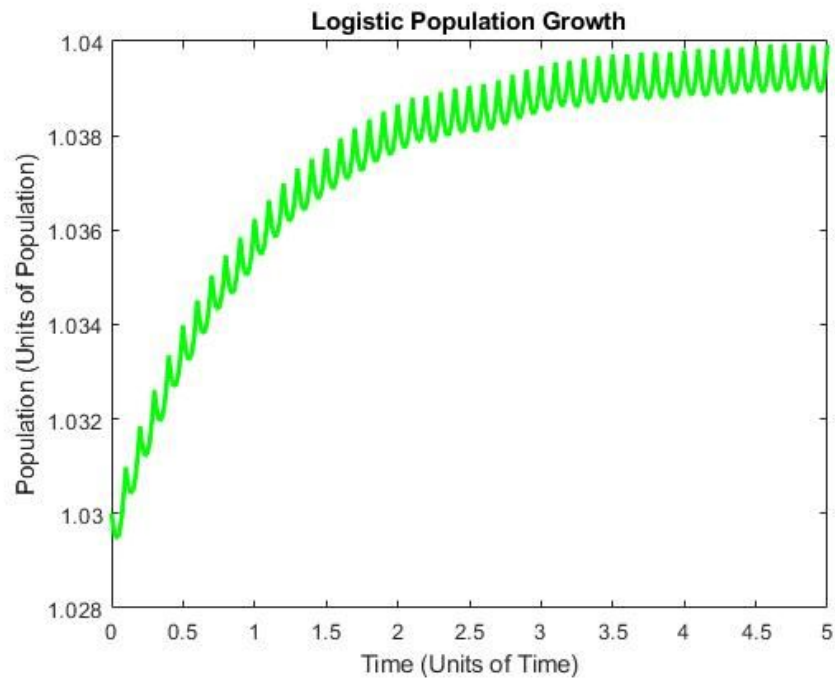


Figure 15: Seasonal model: $tol=1e-6$, $pinit=1.03$, $dur=.1$, $tfin=5$

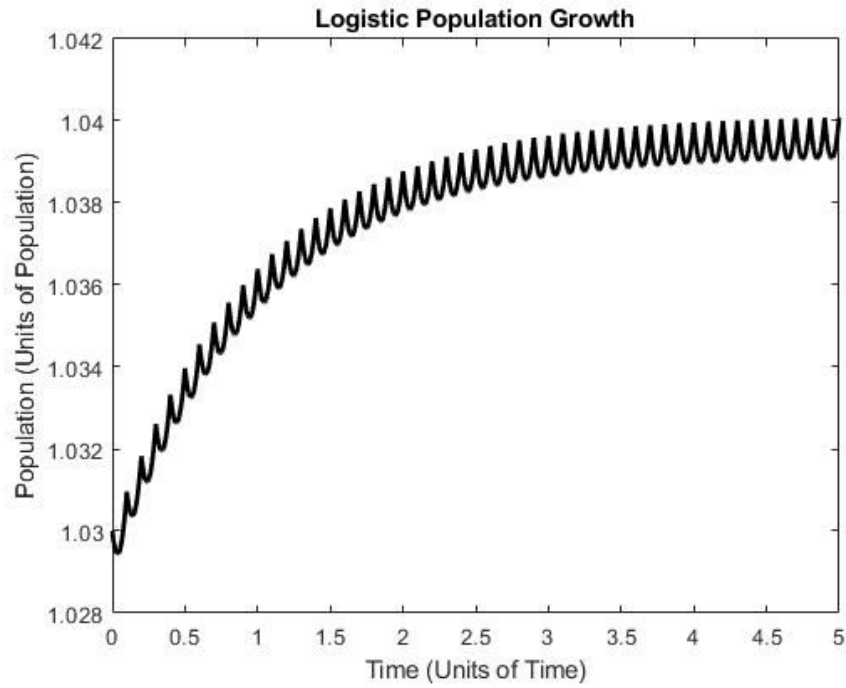


Figure 16: Seasonal model: $tol=1e-12$, $pinit=1.03$, $dur=.1$, $tfin=5$

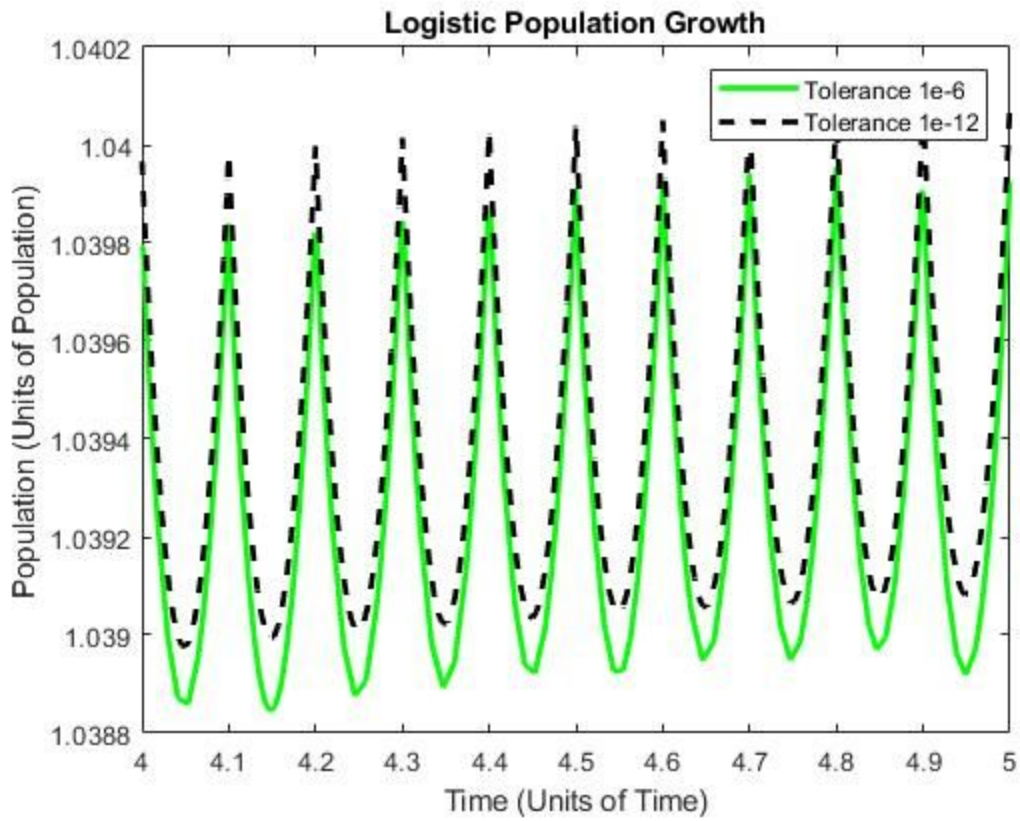


Figure 17: Season model: $tol=1e-6$ and $tol=1e-12$, $pinit=1.03$, $dur=.1$, from $x=4$ to $x=5$

Task 3 Analysis Part 3 (Figures 13-17):

The tolerance $1e-6$ is now inaccurate because the value of dur is much lower than either of the two previous parts (currently .1, was .5 and 2 in parts 1 and 2 of task 3 respectively). This inaccuracy is more pernicious in figure 15 than in figures 13 or 14 because both 13 and 14 are not periodic functions whereas 15 is. This could result in the viewer of the figure assuming that $1e-6$ is accurate enough to properly display the population model as it follows the correct general shape when in reality, figure 16 is the correct model for this seasonal model set of initial conditions and global variables. It can be seen in more detail when looking at figure 17 which clearly shows that the tolerance $1e-6$ is not as accurate as $1e-12$ but still follows the general shape. This implies that figure 16 which has tolerance $1e-12$ is the largest tolerance that can possibly provide an accurate solution for this case.

Task 4: ode15s

Introduction:

The objective of this test was to compare the solve accuracy of *ode15s* to *ode45*. It is noted in the lab instructions that *ode45* is good for engineering computations because it solves quickly.

Ode15s however, often provides a more accurate solution at the cost of computation power and solve time. In order to complete this task, existing code was simply modified from task 3 to run *ode15s* instead of *ode45*. This model was run with two tolerances: $\text{tol}=1.\text{e-}6$ and $\text{tol}=1.\text{e-}12$.

These figures used the same case that produced figure 17.

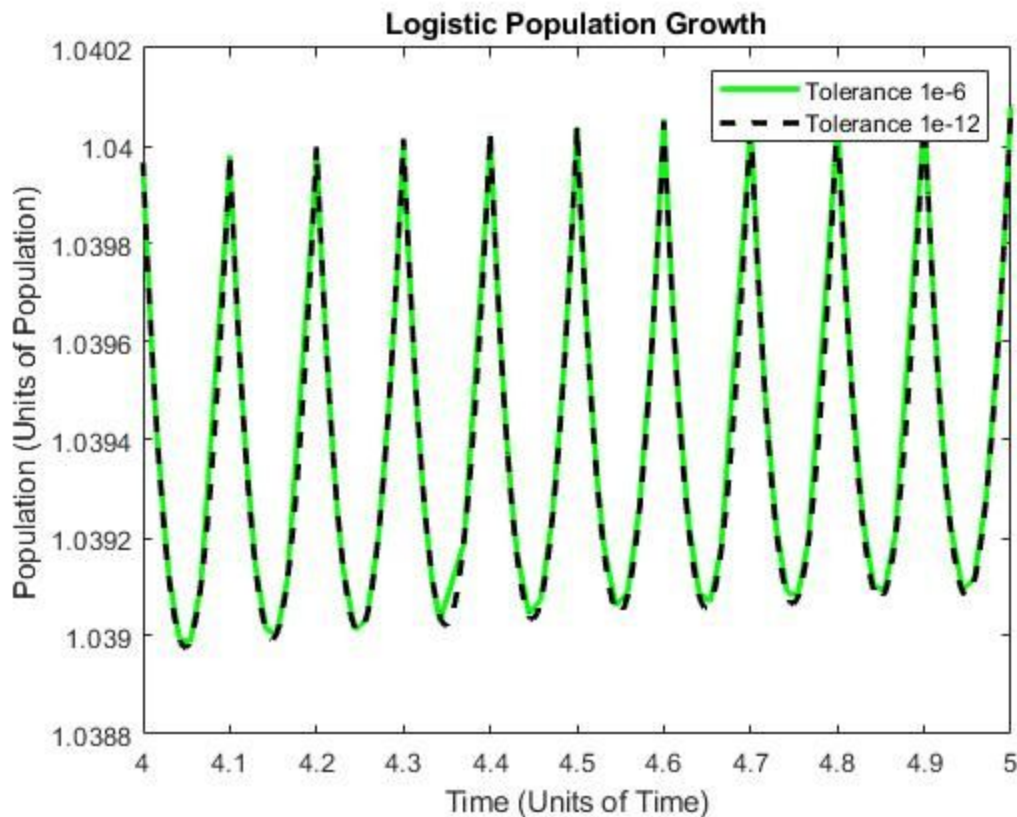


Figure 18: Graph using *ode15s*, tolerances: $1.\text{e-}6$ and $1.\text{e-}12$

Task 4 Analysis

In task 4, the difference between $\text{tol}=1.\text{e-}6$ and $\text{tol}=1.\text{e-}12$ when solving using *ode45* was much larger than the difference between the two tolerances when solved using *ode15s*. This indicates that *ode15s* can achieve much higher levels of solve accuracy with lower tolerances than *ode45*. It should also be noted that both tolerances when solved using *ode15s* produced a curve that seemed to overlap at most points. This indicates that the tolerance $1.\text{e-}6$ eliminated most of the solve error. The case run with $\text{tol}=1.\text{e-}12$ improved the accuracy only marginally compared to $1.\text{e-}6$. Taking the code from task 3, changing the differential equation solver to *ode15s* from

ode45 showed a different result. When comparing the two graphs, the main difference is for the tolerance settings of $1e-6$. Figure 17, the one made using ode45, continues to be less and less precise as time continues whereas when using ode15s shows a line more exact with the lower tolerance of $1e-12$. The goal of the two programs ode15s and ode45 are to computationally solve differential equations. The goal of the two is the same but with different techniques they show different answers given a lower tolerance.

Conclusion & Summary

- In Task 1, the logistics equation was nondimensionalized to make solving feasible using methods outlined above.
- In Task 2, the logistic equation was numerically solved using *ode45* with two different initial populations to represent the effect of carrying capacity on the population. When P_0 was greater than the equilibrium population, the population decreased asymptotically to the carrying capacity. Likewise, when the population was less than the equilibrium population, the population increased asymptotically to the equilibrium population.
- In Task 3, different tolerances were tested to measure the effect of decreasing tolerance on numerical error. In general, it was found that using a smaller tolerance led to a solution that could assumed was closer to the true solution of the equation. As well as tolerance, we looked at how *dur* affected the solution. The figures showed that a larger *dur* was able to create more accurate models with less sporadic jumps.
- In Task 4, the nondimensionalized logistic equation was turned into a nonautonomous differential equation meaning that there was an additional time-based dependency. In particular, *t* affected the carrying capacity of the ecosystem. From the data, it is clear that a seasonal dependence on carry capacity lead to oscillatory behavior as the carrying capacity change with time, so did the population. That is, the population also followed the cyclical nature of carrying capacity.

Appendix 1: Task 2

yprime.m

```
%Creating the function for logistic population growth
function f = yprime(t,p)
f = p*(1 - p);
end
```

pop.m (Figure 1)

```
%Initial Conditions and Setup
tfin = 200;      %Length of time to solve for
pinit = 0.95;    %Initial condition

%Run ode45 for the population equation
[t,p] = ode45(@yprime, [0,tfin],pinit);

%Plot the results and format graph accessories
figure(1);
plot(t,p,'m','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
```

pop.m (Figure 2)

```
%Initial Conditions and Setup
tfin = 200;      %Length of time to solve for
pinit = 1.05;    %Initial condition

%Run ode45 for the population equation
[t,p] = ode45(@yprime, [0,tfin],pinit);

%Plot the results and format graph accessories
figure(2);
plot(t,p,'m','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
```

pop.m (Figure 3)

```
%Initial Conditions and Setup
tfin = 200;      %Length of time to solve for
pinit = 0.95;    %Initial condition

%Setting variables for the different tolerances
tol1 = 1e-4;
tol2 = 1e-6;
```

```

tol3 = 1e-12;

%Run ode45 for the population equation at all 3 new tolerances
options = odeset('RelTol', tol1, 'AbsTol', tol1);
[t,p] = ode45(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode15s(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode15s(@yprime, [0,tfin],pinit,options);

%Plot the results of the new tolerances and format graph accessories
figure(3);
plot(t,p,'r',x,y,'g--',c,f,'k:','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
legend('Tolerance 1e-4','Tolerance 1e-6','Tolerance 1e-12')

```

pop.m (Figure 4)

```

%Initial Conditions and Setup
tfin = 200;      %Length of time to solve for
pinit = 1.05;    %Initial condition

%Setting variables for the different tolerances
tol1 = 1e-4;
tol2 = 1e-6;
tol3 = 1e-12;

%Run ode45 for the population equation at all 3 new tolerances
options = odeset('RelTol', tol1, 'AbsTol', tol1);
[t,p] = ode45(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode15s(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode15s(@yprime, [0,tfin],pinit,options);

%Plot the results of the new tolerances and format graph accessories
figure(4);
plot(t,p,'r',x,y,'g--',c,f,'k:','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
legend('Tolerance 1e-4','Tolerance 1e-6','Tolerance 1e-12')

```


Appendix 2: Task 3

yprime.m

```
%Creating the function for seasonal logistic population growth
function f = yprime(t,p)
global b      %Retrieving global variables
global dur
s = mod(t,dur);
f = p*(1 - (p/(1+(b*s)))));
end
```

pop.m (Figure 5)

```
%Initial Conditions and Setup
tfin = 25;      %Length of time to solve for
pinit = 1.13;   %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.5;

%Run ode45 for the seasonal population equation
[t,p] = ode45(@yprime, [0,tfin],pinit);

%Plot the results and format graph accessories
figure(5);
plot(t,p,'m','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
```

pop.m (Figure 6)

```
%Initial Conditions and Setup
tfin = 25;      %Length of time to solve for
pinit = 1.13;   %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.5;

%Setting variable for the new tolerance
tol1 = 1e-4;

%Run ode45 for the seasonal population equation at given tolerance
options = odeset('RelTol', tol1, 'AbsTol', tol1);
```

```
[t,p] = ode45(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(6);
plot(t,p,'r','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
```

pop.m (Figure 7)

```
%Initial Conditions and Setup
tfin = 25;          %Length of time to solve for
pinit = 1.13;      %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.5;

%Setting variables for the different tolerances
tol2 = 1e-6;
tol3 = 1e-12;

%Run ode45 for the seasonal population equation at given tolerances
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode15s(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode15s(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(7);
plot(x,y,'g',c,f,'k--','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
legend('Tolerance 1e-6','Tolerance 1e-12')
```

pop.m (Figure 8)

```
%Initial Conditions and Setup
tfin = 25;          %Length of time to solve for
pinit = 1.13;      %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.5;
```

```

%Setting variables for the different tolerances
tol2 = 1e-6;
tol3 = 1e-12;

%Run ode45 for the seasonal population equation at given tolerances
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode15s(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode15s(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(8);
plot(x,y,'g',c,f,'k--','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
legend('Tolerance 1e-6','Tolerance 1e-12')

%Zooming between times 20 and 25
xlim([20 25])

```

pop.m (Figure 9)

```

%setting the ending time and the initial population
tfin = 25;
pinit = 1.13;
%adding the global variable b
global b
b = 0.8;
%adding the global variable dur
global dur
dur = 2;
[t,p] = ode45(@yprime, [0,tfin],pinit);
figure(9);
%plotting the population
plot(t,p,'m','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')

```

pop.m (Figure 10)

```

%setting the ending time and the initial population
tfin = 25;
pinit = 1.13;
%adding the global variable b
global b
b = 0.8;
%adding the global variable dur
global dur
dur = 2;

```

```

%denoting the new tolerance
tol2 = 1.e-4;
%using the command odeset, change the tolerance to the one specified
%earlier in the code (tol2)
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode45(@yprime, [0,tfin],pinit,options);
figure(10);
%plotting the new tolerance
plot(x,y,'r','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
%labeling the the population by adding a legend
legend('Tolerance 1e-4')

```

pop.m (Figure 11)

```

%setting the ending time and the initial population
tfin = 25;
pinit = 1.13;
%adding the global variable b
global b
b = 0.8;
%adding the global variable dur
global dur
dur = 2;
%denoting the new tolerances
tol2 = 1.e-6;
tol3 = 1e-12;
%using the command odeset , change the tolerances to the ones specified
%earlier in the code (tol2 and tol3)
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode45(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode45(@yprime, [0,tfin],pinit,options);
figure(11);
%plotting the two tolerances
plot(x,y,'g',c,f,'k--','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
%labeling the two tolerances by color by adding a legend
legend('Tolerance 1e-6','Tolerance 1e-12')

```

pop.m (Figure 12)

```

%setting the ending time and the initial population
tfin = 25;
pinit = 1.13;
%adding the global variable b
global b

```

```

b = 0.8;
%adding the global variable dur
global dur
dur = 2;
%denoting the new tolerances
tol2 = 1.e-6;
tol3 = 1e-12;
%using the command odeset, change the tolerances to the ones specified
%earlier in the code (tol2 and tol3)
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode45(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode45(@yprime, [0,tfin],pinit,options);
figure(12);
%plotting the two tolerances
plot(x,y,'g',c,f,'k--','LineWidth',2.0)
%setting the range of x shown from 20 to 25
xlim([20 25]);
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
%labeling the two tolerances by color by adding a legend
legend('Tolerance 1e-6','Tolerance 1e-12')

```

pop.m (Figure 13)

```

%Initial Conditions and Setup
tfin = 5;          %Length of time to solve for
pinit = 1.03;      %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.1;

%Run ode45 for the seasonal population equation
[t,p] = ode45(@yprime, [0,tfin],pinit);

%Plot the results and format graph accessories
figure(13);
plot(t,p,'m','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')

```

pop.m (Figure 14)

```

%Initial Conditions and Setup
tfin = 5;          %Length of time to solve for
pinit = 1.03;      %Initial condition

%Adding the global variables b and dur

```

```

global b
b = 0.8;
global dur
dur = 0.1;

%Setting variable for the new tolerance
tol1 = 1e-4;

%Run ode45 for the seasonal population equation at given tolerance
options = odeset('RelTol', tol1, 'AbsTol', tol1);
[t,p] = ode45(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(14);
plot(t,p,'r','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')

```

pop.m (Figure 15)

```

%Initial Conditions and Setup
tfin = 5;          %Length of time to solve for
pinit = 1.03;      %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.1;

%Setting variable for the new tolerance
tol2 = 1e-6;

%Run ode45 for the seasonal population equation at given tolerance
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode45(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(15);
plot(x,y,'g','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')

```

pop.m (Figure 16)

```

%Initial Conditions and Setup
tfin = 5;          %Length of time to solve for
pinit = 1.03;      %Initial condition

```

```

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.1;

%Setting variable for the new tolerance
tol3 = 1e-12;

%Run ode45 for the seasonal population equation at given tolerance
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode45(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(16);
plot(c,f,'k','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')

```

pop.m (Figure 17)

```

%Initial Conditions and Setup
tfin = 5;          %Length of time to solve for
pinit = 1.03;      %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.1;

%Setting variables for the new tolerances
tol2 = 1e-6
tol3 = 1e-12;

%Run ode45 for the seasonal population equation at given tolerances
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode45(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode45(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(17);
plot(x,y,'g',c,f,'k--','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
legend('Tolerance 1e-6','Tolerance 1e-12')

```

```
%Zooming between times 4 and 5  
xlim([4 5])
```


Appendix 3: Task 4

yprime.m

```
%Creating the function for seasonal logistic population growth
function f = yprime(t,p)
global b      %Retrieving global variables
global dur
s = mod(t,dur);
f = p*(1 - (p/(1+(b*s)))));
end
```

pop.m (Figure 18)

```
%Initial Conditions and Setup
tfin = 5;      %Length of time to solve for
pinit = 1.03;  %Initial condition

%Adding the global variables b and dur
global b
b = 0.8;
global dur
dur = 0.1;

%Setting variables for the new tolerances
tol2 = 1e-6
tol3 = 1e-12;

%Run ode15s for the seasonal population equation at given tolerances
options = odeset('RelTol', tol2, 'AbsTol', tol2);
[x,y] = ode15s(@yprime, [0,tfin],pinit,options);
options = odeset('RelTol', tol3, 'AbsTol', tol3);
[c,f] = ode15s(@yprime, [0,tfin],pinit,options);

%Plot the results and format graph accessories
figure(18);
plot(x,y,'g',c,f,'k--','LineWidth',2.0)
xlabel('Time (Units of Time)')
ylabel('Population (Units of Population)')
title('Logistic Population Growth')
legend('Tolerance 1e-6','Tolerance 1e-12')

%Zooming between times 4 and 5
xlim([4 5])
```