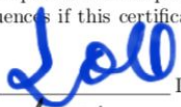





Lab Report 2

Team Members: Joie Green, Melissa Prado, Niva Razin, Alexander Ross

EA4 Lab Certification

We certify that the members of the team named below worked on this lab using only our own ideas, possibly with 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. Name	Joie Green	Signature		Date	1/29/2020
2. Name	Melissa Prado	Signature		Date	1/29/2020
3. Name	Niva Razin	Signature		Date	1/29/2020
4. Name	Alexander Ross	Signature		Date	1/29/2020

INTRODUCTION

In this lab, we applied the Improved Euler (IE) method to population models. We developed a differential equation solver in MATLAB based on the IE method and implemented it to better understand and analyze the behavior of critical points and bifurcation points. Specifically, we implemented an adaptive and non-adaptive solver to computationally analyze our population model. This process enhanced our understanding of the crowding effect, and how it impacts population models with regards to doomsday, critical points, and bifurcation points. In turn, we gained valuable experience in MATLAB coding, software development, mathematical modeling, critical thinking/analysis, writing, collaboration, and project management.

TASK 1

We begin by using a population model that describes a species with a negative natural birth rate:

$$\frac{dP}{dt} = PN(P) = P(-1 + P - \alpha P^2) \quad (1)$$

We set $\alpha = 0$, thus eliminating the term which corrects for the crowding effect:

$$\frac{dP}{dt} = P(-1 + P) \quad (2)$$

By definition, critical points are given by solutions to $\frac{dP}{dt} = 0$. We set (2) equal to zero and solve:

$$\frac{dP}{dt} = P(-1 + P) = 0$$

$$P = 0, (-1 + P) = 0 \rightarrow P = 1$$

We get two solutions, $P_{ext} = 0$ and $P_{middle} = 1$. We use the derivative test to find the stability of these critical points: we expand (2), set it equal to $f(P)$, find $\frac{df}{dP}$, and plug in P_{ext} and P_{middle} :

$$\frac{dP}{dt} = P(-1 + P) = -P + P^2 = f(P)$$

$$\frac{df}{dP} = -1 + 2P$$

$$\left. \frac{df}{dP} \right|_{P_{ext}} = -1 + 2(0) = -1 < 0$$

$$\left. \frac{df}{dP} \right|_{P_{middle}} = -1 + 2(1) = 1 > 0$$

The derivative test for P_{ext} results in a negative number, indicating it is a stable critical point. Conversely, the derivative test for P_{middle} results in a positive number, indicating it is an unstable critical point.

Now, we solve (2) using the partial fraction integration technique:

$$\frac{dP}{dt} = P(-1 + P) \rightarrow \int \frac{1}{P(-1+P)} dP = \int dt$$

$$\frac{A}{P} + \frac{B}{(-1+P)} = \frac{A(-1+P) \times B(P)}{P(-1+P)} = \frac{1}{P(-1+P)} \rightarrow A(-1 + P) \times B(P) = 1$$

We set P to an arbitrary number to solve for A and B :

$$P = 0 \rightarrow A(-1 + 0) \times B(0) = 1 = -A = 1 \rightarrow A = -1$$

$$P = 1 \rightarrow A(-1 + 1) \times B(1) = 1 = B = 1 \rightarrow B = 1$$

We use these values to continue integration:

$$\int \frac{1}{P} + \int \frac{1}{(-1+P)} = \int dt$$

$$- \ln |P| + \ln |-1 + P| = \ln |(-1 + P)/P| = t + C$$

We raise both sides to e :

$$\frac{(-1+P)}{P} = e^{t+C} = C e^t \quad (3)$$

We plug in the initial condition $P(0) = P_0$ to solve for C :

$$\frac{(-1+P_0)}{P_0} = C e^0 = C$$

We substitute this expression for C back into (3):

$$\frac{(-1+P)}{P} = \left(\frac{(-1+P_0)}{P_0} \right) e^t$$

We cross multiply and simplify to find an expression for $P(t)$:

$$P(t) = \frac{P_0}{P_0 - (P_0 - 1)e^t} \quad (4)$$

Next, we evaluate (4) for two possible ranges of physical initial population P_0 values. First, when $P_0 > P_{middle}$, the denominator of (4) approaches zero as t increases to a certain point, $t_{doomsday}$, causing $P(t) \rightarrow \infty$. Second, when $0 < P_0 < P_{middle}$, the denominator of (4) approaches infinity as $t \rightarrow \infty$, causing $P(t) \rightarrow 0$.

We now consider the first case, the unphysical situation called *doomsday*, in which the population grows to infinity as time goes on. In mathematical terms, doomsday is when $P(t) \rightarrow \infty$ as $t \rightarrow t_{doomsday}$. We can set the denominator of (4) equal to zero to solve for the time $t_{doomsday}$:

$$P_0 - (P_0 - 1)e^t = 0 \rightarrow e^t = \frac{P_0}{(P_0 - 1)} \rightarrow t_{doomsday} = \ln \left(\frac{P_0}{(P_0 - 1)} \right)$$

Note that this $t_{doomsday}$ only exists for $P_0 > P_{middle} = 1$. This is because when

$P_{ext} < P_0 < P_{middle}$, the denominator of (4) is always positive, and thus $P(t) \rightarrow 0$ as $t \rightarrow \infty$. Thus, P_{middle} acts as a threshold value in this model. That is, when $P_0 > P_{middle}$, the

population always progresses to infinity, and when $P_{ext} < P_0 < P_{middle}$, the population always progresses to extinction.

TASK 2

In this task we write two programs based on the Improved Euler method. The program *fixed.m* performs the Improved Euler method with a fixed timestep while the *adapt.m* program performs the method with a varying timestep.

The *fixed.m* program begins by calculating the number of times Euler's formula should run for a fixed timestep h . Then the program calculates t from 0 to t_{fin} and stores them inside a .mat file. If the final t value is greater than t_{fin} , then the function changes the final t to t_{fin} . In another program, *matlabtask2fixed.m*, the stored t values are loaded from the .mat file. Then the program calculates y values using Euler's formula. The derivative used in Euler's formula at the value t is calculated by the function *yprime*, which is called by *matlabtaskfixed.m*.

The *adapt.m* program begins by reading in initial values from a separate data file named *adapt.dat*. The program then goes through a while loop meant to determine if the timestep h is sufficiently small; if the timestep is too large, the program divides h by a set value of $hcut$ in order to lower the value. Once the timestep is sufficiently small, the program then runs another while loop that performs the Improved Euler method.

The derivative at the current value of y , y_{init} , is calculated and then used to find the two correctors, y_{low} and y_{high} , as well as the current value of time t . A logical expression is then used to adjust the timestep, depending on whether or not the relative error is greater than the set tolerance, ϵ . If the relative error is larger than ϵ , then the timestep is cut further. Otherwise, the new values for t and y_{init} are set for the next loop, the count is updated, and h is raised by a factor of the previously set value $hraise$.

Both programs are then run independently with various values of h and ϵ , and the results are shown below in Table 1.

Table 1: Comparison of *adapt.m* and *fixed.m* with $\alpha = 0$.

M	h	ϵ	count	ycomp	yex	err	rerr
A		0.00005	33524	347.6862	348.8844	0.4200	0.0012
F	0.00018		1583	273.9233	348.1061	74.1829	0.2131
A		0.0001	31593	348.8844		0.7782	0.0022
F	0.00025		1140	254.2916	348.1061	93.8146	0.2695

A		0.0005	34149	347.2260		0.8802	0.0025
F	0.0006		475	191.3957	348.1061	156.7105	0.4502
A		0.001	34592	341.7223		6.3839	0.0268
F	0.0008		357	168.9891	348.1061	179.1171	0.5145

The final value of count for each run is representative of the work required to obtain a solution. This is because each iteration of the solver adds to the count, and the equation solver runs until the desired time is reached.

The results for the two most accurate runs of each solver are plotted in **Figure 1** and **Figure 2**

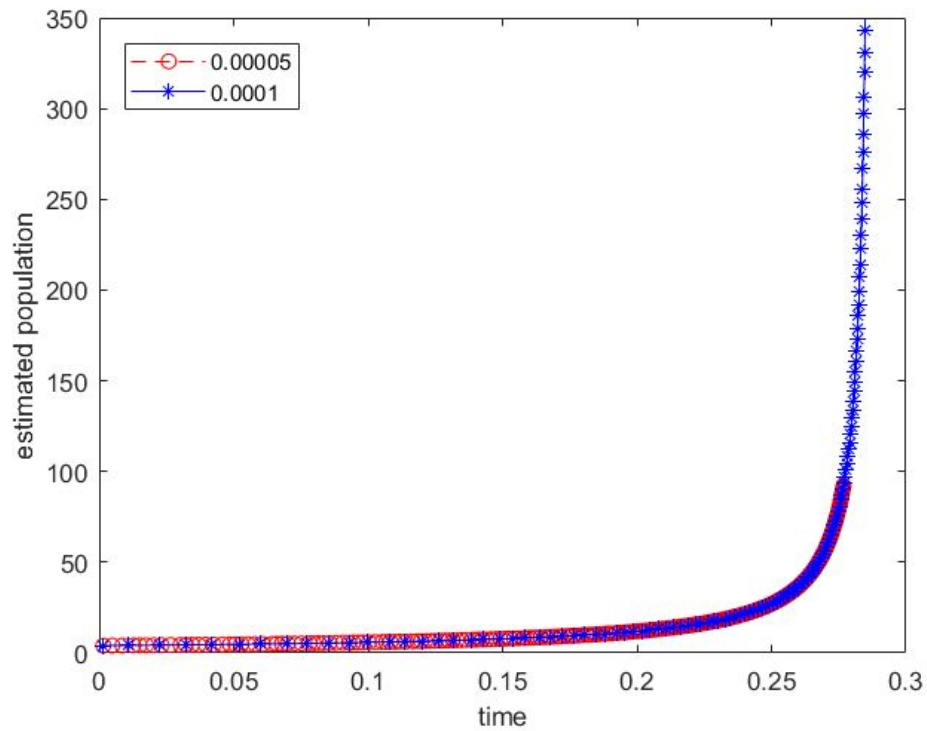


Figure 1: Graphs the adaptive solution with $\epsilon=0.00005$ and the $\epsilon=0.0001$

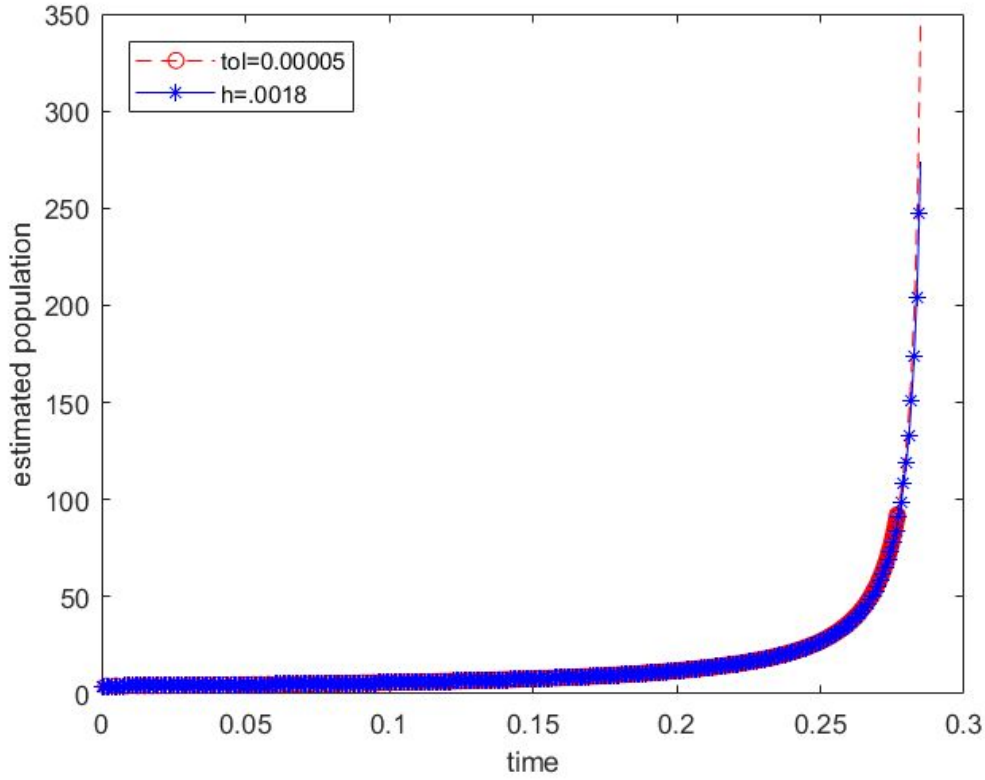


Figure 2: graphs the adaptive solution with $\epsilon > 0.00005$ and the fixed solution with timestep $h=0.0018$

Figure 1 and **Figure 2** show that the highest resolution computations are trustworthy because they are nearly identical both quantitatively and qualitatively. The two curves are right on top of each other in both figures, which shows that the results are highly accurate.

TASK 3

We now solve for the critical points of (1) when $\alpha > 0$:

$$\frac{dP}{dt} = P(-1 + P - \alpha P^2) = 0$$

By observation, we see that one solution of $\frac{dP}{dt} = 0$ is given by the critical point $P_{ext} = 0$. We then solve the quadratic expression of (1):

$$-1 + P - \alpha P^2 = 0 = 1 - P + \alpha P^2 \quad (5)$$

Plugging into the quadratic formula, we solve for the roots of (5), which are also two more critical points of (1) because they satisfy $\frac{dP}{dt} = 0$:

$$P = \frac{-(-1) \pm \sqrt{(1)^2 - 4(\alpha)(1)}}{2(\alpha)} = \frac{1 \pm \sqrt{1-4\alpha}}{2\alpha} \rightarrow P_{big} = \frac{1 + \sqrt{1-4\alpha}}{2\alpha}, P_{middle} = \frac{1 - \sqrt{1-4\alpha}}{2\alpha}$$

Since we know that $\alpha > 0$, we can tell that the critical point $P_{big} > P_{middle}$. Additionally, because $P_{ext} = 0$ is the lowest possible physical value of P , we know that $P_{ext} < P_{middle} < P_{big}$.

In order for P_{big} and P_{middle} to have real values and be physical critical points, the term under their square root term must be positive:

$$1 - 4\alpha > 0 \rightarrow \alpha < \frac{1}{4}$$

In turn, we see that $0 < \alpha < \alpha_*$, where $\alpha_* = \frac{1}{4}$.

We again use the derivative test to find the stability of the first critical point: we expand (1), set it equal to $f(P)$, find $\frac{df}{dP}$, plug in P_{ext} , and solve. Note that this $f(P)$ is different than that used in Task 1:

$$\frac{dP}{dt} = P(-1 + P - \alpha P^2) = -P + P^2 - \alpha P^3 = f(P) \quad (6)$$

$$\frac{df}{dP} = -1 + 2P - 3\alpha P^2 \quad (7)$$

$$\left. \frac{df}{dP} \right|_{P_{ext}} = -1 + 2(0) - 3\alpha(0)^2 = -1 < 0$$

For P_{big} and P_{middle} , we will instead use a phase diagram to evaluate the behavior of $f(P)$ at arbitrary values of P before and after each critical point. Since we know that these critical points are also roots of the quadratic expression of (5), we can rewrite $f(P)$ as:

$$f(P) = -P(P - P_{middle})(P - P_{big})$$

Given that P_{big} and P_{middle} must be positive in order to be physical, we can evaluate the following signs:

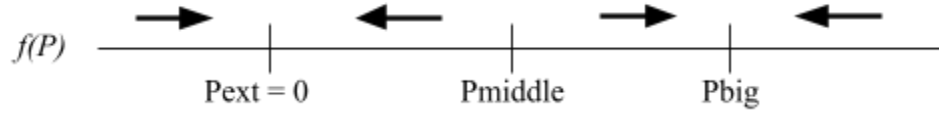
$$f(-\infty) = (+)(-)(-) = (+)$$

$$f(P_{ext} < P < P_{middle}) = (-)(-)(-) = (-)$$

$$f(P_{middle} < P < P_{big}) = (-)(+)(-) = (+)$$

$$f(+\infty) = (-)(+)(+) = (-)$$

These signs are indicated by the arrows on the phase diagram below. We see that for all values of α , P_{ext} and P_{big} are stable, as $f(P)$ converges at these critical points. P_{middle} , however, is unstable, as $f(P)$ diverges from this critical point.



Now, we evaluate P_{middle} and P_{big} when $\alpha = \alpha_* = \frac{1}{4}$:

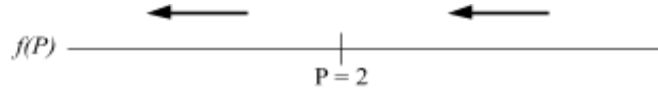
$$P_{big} = \frac{1 + \sqrt{1 - 4(\frac{1}{4})}}{2(\frac{1}{4})} = \frac{1+0}{(\frac{1}{2})} = 2$$

$$P_{middle} = \frac{1 - \sqrt{1 - 4(\frac{1}{4})}}{2(\frac{1}{4})} = \frac{1-0}{(\frac{1}{2})} = 2$$

We see that P_{middle} and P_{big} merge at the critical point $P_* = 2$ when $\alpha = \alpha_*$ (**Figure 3**). We again perform the derivative test, this time plugging in the critical point $P = 2$ and $\alpha = \frac{1}{4}$ into (7):

$$\left. \frac{df}{dP} \right|_P = -1 + 2(2) - 3\left(\frac{1}{4}\right)(2)^2 = -1 + 4 - 3 = 3 - 3 = 0$$

The result is zero, indicating that P_* is semistable and that we must use the sign test to determine its stability:



When we evaluate $f(P)$ at an arbitrary $0 < P < 2$ and a $P > 2$, we get negative numbers. This observation is shown by the leftward arrows in the phase diagram above. We see that P_* is stable from above (i.e., $f(P)$ converges on the critical point from above) and unstable from below (i.e., $f(P)$ diverges from the critical point from below).

When $\alpha < \alpha_*$, there are three critical points, $P_{ext} = 0$, P_{middle} , and P_{big} . The two stable critical points P_{big} and P_{ext} indicate the population's behavior at extremes, when it has stabilized at its carrying capacity or gone extinct, respectively. Depending on P_0 , the unstable critical point P_{middle} determines whether the population progresses to P_{big} or P_{ext} . When $\alpha > \alpha_*$, the square root terms of P_{middle} and P_{big} result in complex numbers with no significance to population modeling; only the critical point $P_{ext} = 0$ exists. This change in the

number of critical points occurs at the bifurcation point, when $\alpha = \alpha_*$. At the bifurcation point, there are two critical points, $P_* = 2$ and $P_{ext} = 0$. **Figure 3** shows this behavior.

We will now consider the behavior of the critical points P_{middle} and P_{big} as $\alpha \rightarrow 0$ from above, similar to the behavior analyzed in Task 1. For P_{big} , we can substitute 0 in for α in the numerator because the α here affects the value of P_{big} far less than the α in the denominator. The fractional values of alpha only slightly decrease the value of the numerator, though they significantly decrease the value of the denominator:

$$P_{big} \approx \frac{1 + \sqrt{1 - 4(0)}}{2\alpha} \approx \frac{1 + \sqrt{1}}{2\alpha} \approx \frac{2}{2\alpha} \approx \frac{1}{\alpha}$$

Thus, as $\alpha \rightarrow 0$, $P_{big} \rightarrow \infty$.

For P_{middle} , we use the $P_{middle} = 1$ found in Task 1 when $\alpha = 0$ to approximate:

$$P_{middle} \approx P_{approximation_{middle}(\alpha)} \approx 1 + \alpha v$$

We then plug this into (6):

$$f(1 + \alpha v) = -(1 + \alpha v) + (1 + \alpha v)^2 - \alpha(1 + \alpha v)^3 = 0$$

We use the fact that when α is small, $\alpha^3 \ll \alpha^2 \ll \alpha$ to keep only constants and terms proportional to α to find v :

$$\alpha v - \alpha = 0 \rightarrow \alpha v = \alpha \rightarrow v = 1$$

This value of v then gives $P_{middle} \approx 1 + \alpha$, and as $\alpha \rightarrow 0$, we see that $P_{middle} \rightarrow 1$.

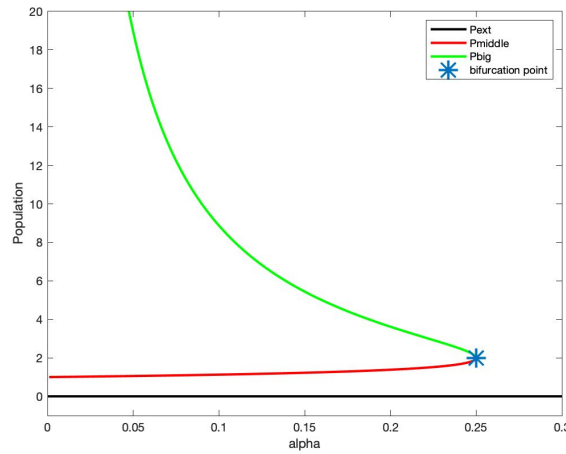


Figure 3: Plots of the three critical points of equation (1) for various α . Past the bifurcation point only P_{ext} exists.

TASK 4

In Task 4, the program *adapt.m* was used to generate several population models where the initial population count of each model was varied. From this, we were able to generate a graph of how each population varied with respect to time, and we drew conclusions about how population growth changes with different initial population counts. The results can be seen in **Figure 4**

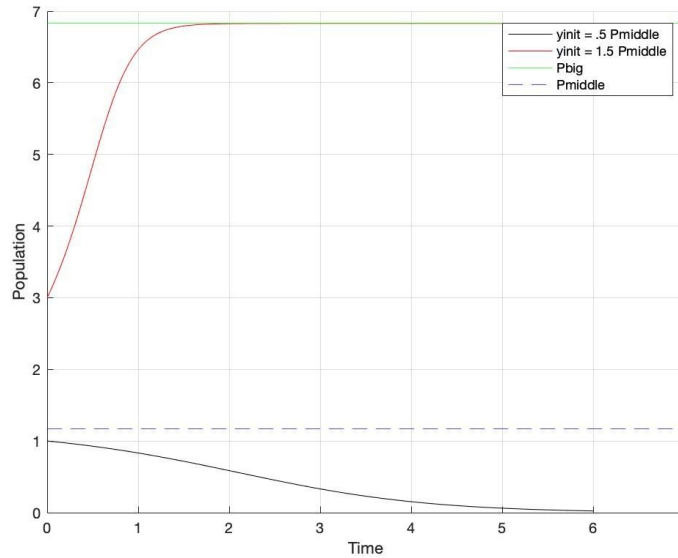


Figure 4. Plot of the solutions of *adapt.m* for various $y_{initial}$ values. P_{big} and P_{middle} are plotted for reference at the desired value of α ($\alpha = 1/8$)

Figure 4 shows how the size of the initial population (Y_{init}) determines the growth pattern that the population will follow. That is, when P_0 is less than P_{middle} , the population converges to the stable critical point $P_{ext} = 0$ (i.e., population goes extinct). However, when the population begins at a size between P_{middle} and P_{big} , the population converges on the P_{big} critical point. This model corrects for the unrealistic doomsday scenario where the population grows infinitely when $P_0 > P_{middle}$. Instead, it shows the more realistic behavior that as the population grows, it eventually stabilizes at its carrying capacity P_{big} , and the crowding effect thus limits the growth of the population. Ergo, the results of this plot agree with our analysis in Task 3.

CONCLUSION

The primary focus of this lab was using MATLAB to develop an adaptive differential equation solver, and implement it to solve various problems in population modeling. Task 2 was devoted to developing and testing this MATLAB code. By utilizing adaptive methods, we were able to effectively optimize how well we could model a differential equation computationally. We learned how to design adaptive procedures for numerical analysis, as well as how to test our program to ensure it was working properly. This set the stage for the numerical analysis of our population models.

We then learned about the nature of critical and bifurcation points, as well as how they can vary in population models. In Tasks 1 and 3, we determined mathematical solutions for critical and bifurcation points of the non-dimensionalized population model. We considered how different values of α in our crowding term (αP^2) would affect the critical points in the model, and how unstable critical points could be interpreted as threshold values. This gave us a solid mathematical understanding of critical and bifurcation points, allowing us to make computational assessments of our models using our adaptive equation solver.

When applied, all of our previous work contributed to the final analysis of our population models. Task 4 involved interpreting our MATLAB results in terms of the behavior of population models with negative birth rates. We saw that the nature of population growth changes based on where the initial population is with respect to the threshold value. Throughout this process, we saw that the doomsday versus extinction model was unphysical and how the crowding term eliminates this to ensure that our model remains physical. Thus, we have developed a more complete understanding and appreciation for the intricacies of population modeling. Finally, this lab afforded us the opportunity to work in a group. With this opportunity came the challenge of effective communication, equitable delegation of tasks, and time management.

Appendix A:

Adapt:

%Set initial conditions

global alpha

global count

count2 = 0;

[~,~,h,~,~,hraise,hbig,hcut,hsmall]= textread('adapt.dat','%f %f %f %d %f %f %d %d %f',1);

alpha= .0

yinit= 4

epsilon=0.0001

t=0;

tdoomsday = (log(4/(4-1)));

tfin = .99*tdoomsday

ma(1,1)=0

ma(1,2)=yinit %Improved Euler loop

while t+h>= tfin %cuts initial h so second loop can run

h=h/hcut

End

while t<=tfin %runs when h is sufficiently small

%Go through IE Method while saving t value

q= yprime(yinit) %derivative at yinit

y1 = yinit +(h*q); %y1low(euler's formula)

p=yprime(y1);

tempy = yinit+ (h/2)*(q+p); %yhigh euler corrector

```

tempt = t+h; %temporary t

%If difference between y1 and tempy divided by y1 is larger than
%tolerance, h is adjusted until value is smaller than tolerance
k=abs(y1-tempy)/abs(y1); %calculates relative error
    if k>=epsilon %if error is larger than our tolerance
        h = h/hcut;
        if h<hsmall %checks if new h is too small
            error('Step size is too small')%Give error message if h is too small
            %Set new values of t and yinit for next loop if there is no error
        Else
            End
        Else
t=tempt;
yinit=tempy;
count2= count2+1
ma(count2+1,1)=t;
ma(count2+1,2)=yinit;
h= h*hraise; %raise h by factor of hraise
end
End
    if t>tfin
t=tfin
ma(count2+1,1)=t;
yinit= ma(count2,2)
y1 = yinit +(h*q); %y1low(euler's formula)

```

```

p=yprime(y1);
tempy = yinit+ (h/2)*(q+p);
ma(count2+1,2)= tempy;
Else
end
yex = 4/(4-(4-1)*exp(tfin))
tempy
errfin = abs(tempy - yex)
rerr = errfin/yex

```

Yprime:

```

function f= yprime(t) %Set up global variables & set up function formula
    global alpha
    f = t*(-1+t-alpha*(t^2));%Set up function formula
    global count %Keep track of number of times function runs
    Count = count+1;
End

```

Fixed:

```

t=0;% initial t value
h= .0008, %change in time in each step
yinit= 4; % P when t=0 initial
a=0
tdoom= log(yinit/(yinit-1)); %determines t-doomsday
tfin=.99*tdoom;%calculated t fin from tdoosmday
rows= tfin/h; % determines number of times euler's formula should run based time step and tfin

```



```

go= ceil(rows) + 1; %rounds up to the next integer
n(1,1)= 0; %sets first value in t matrix to 0
yex= yinit/(yinit-((yinit-1)*exp(tfin))) %calculates exact solution at tfin
for i = 1:go %for loop
    t= t+h %calculates time after adding change in t
    if t>= tfin %if calculated t value is greater than or equal to tfin then
        n(i+1,1)=tfin
        break %end the for loop
    else
        n(i+1,1)= t; % if t value is less than tfin then continue with for loop
    end
end
End

```

Matlabtask2fixed.m

```

f= matfile('1-tol-lab2'); %% loads .mat data file into f variable
u=f.n; % calls t-data matrix of above file and stores into variable u
y= matfile('yex'); %calls variable yex
yex= y.yex; %sets yex equal to yex(calculated in fixed.m)
a= f.a;
u(1,2)= f.yinit; %sets P=4 when t=0 in u matrix
for i= 2:length(u) %for loop
    dpdt= yprime2(u(i-1,2),a); %calculated dpdt using yprime- yprime gets inputs of u matrix
    h=u(i,1)-u((i-1),1);
    y=u(i-1,2)+ (h*dpdt); %euler's formula step
    u(i,2)=y; %puts calculated value into the 2nd column of u matrix(which contains t values)
end

```

count

y %equivalent to y comp - prints value at tfin

errfin= abs(y-yex) %absolute error of the computation

reer= errfin/yex %relative error of the computation