


**EA4 Lab 2:**

10/15/19

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 2: Thermal Explosion

Exothermic chemical reactions can cause thermal explosions from organic materials under certain circumstances, which poses a big hazard when storing and shipping these materials. In this lab a simple model of a thermal explosion will be used to visualize how the temperature of organic materials can change in order to reach a state of equilibrium within its system (i.e. when heat generation balances heat loss). MATLAB will be used to numerically evaluate the model for thermal explosion through the implementation of the improved Euler method with both fixed and adaptive step sizes. Over the course of the lab, the improved Euler method, and the thermal explosion equation will be modeled and studied to measure the affects the numerical solution on the precision of a differential equation model.

## Task 1:

### Introduction:

The goal of Task 1 is to nondimensionalize the thermodynamic model that is the focus of this lab. The first step is changing the variables and reducing the equation. This process will make solving the differential equation much easier than if the equation depended on many parameters. Another advantage to reducing the number of parameters is reducing the complexity of the equation and reducing the introduction of variable errors into the solution of the differential equation.

### Task 1:

The initial value problem is given to model a thermal explosion:

$$mc \frac{d\hat{T}}{dt} = BQe^{-\frac{E}{RT}} - hS(\hat{T} - \hat{T}_s), \quad \hat{T}(0) = \hat{T}_0$$

To simplify this model, add these nondimensional variables:

$$T = \frac{\hat{T}}{\hat{T}_s} \text{ and } \tilde{t} = \hat{t}B$$

Replacing the original variables with the nondimensional variables, to get a new equation:

$$\Rightarrow mcB\hat{T}_s \frac{dT}{d\tilde{t}} = BQe^{-\frac{E}{RT_s T}} - hS(\hat{T}_s T - \hat{T}_s)$$

which can then be reduced to:

$$\Rightarrow \frac{dT}{d\tilde{t}} = \frac{BQe^{-\frac{E}{RT_s T}} - hS\hat{T}_s(T-1)}{mcB\hat{T}_s} = \frac{Q}{mc\hat{T}_s} e^{-\frac{E}{RT_s T}} - \frac{hS}{mcB}(T-1)$$

To simplify the equation further, a change of variables is made where some variables are grouped into 1 variable in three different occasions:

$$\text{Let } D = \frac{Q}{mc\hat{T}_s}, \quad \gamma = \frac{E}{RT_s T}, \quad k = \frac{hS}{mcB}$$

Now with these definitions one can arrive at our final equation:

$$\Rightarrow \frac{dT}{d\tilde{t}} = De^{-\frac{\gamma}{T}} - k(T-1)$$

Using the original equation for the nondimensional variable T, one can find what T<sub>0</sub> is equal to:

$$T = \frac{\hat{T}}{\hat{T}_s} \Rightarrow \text{at } t = 0 \dots T_0 = \frac{\hat{T}_0}{\hat{T}_s}$$

and since this is a nondimensional variable, T<sub>0</sub> will have no units.

To find the units for the three other variables created we'll have to see how the variables relate to the original parameters of the problem. For reference: J=Joules, kg=kilograms, K=Kelvin, mol=moles, s=seconds, m=meters. Let's start with D and its parameters:

$$D = \frac{Q}{mc\hat{T}_s} \Rightarrow Q = \frac{J}{1}, \frac{1}{m} = \frac{1}{kg}, \frac{1}{c} = \frac{K}{J}, \text{ and } \frac{1}{\hat{T}_s} = \frac{1}{K}$$

When combining the units for all the parameters one get the units for D:

$$D = \left(\frac{J}{1}\right)\left(\frac{1}{kg}\right)\left(\frac{K}{J}\right)\left(\frac{1}{K}\right) = \frac{1}{kg}$$

Next let's look at  $\gamma$  and the units for its parameters:

$$\gamma = \frac{E}{R\hat{T}_sT} \Rightarrow E = \frac{J}{mol}, \frac{1}{R} = \frac{mol*K}{J}, \text{ and } \frac{1}{\hat{T}_s} = \frac{1}{K}$$

Combining the units for all the parameters we get the units for  $\gamma$ :

$$\gamma = \left(\frac{J}{mol}\right)\left(\frac{mol*K}{J}\right)\left(\frac{1}{K}\right) = \text{No Units}$$

Finally, let's look at k and the units for its parameters:

$$k = \frac{hS}{mcB} \Rightarrow h = \frac{J}{s * m^2 * K}, S = \frac{m^2}{1}, \frac{1}{m} = \frac{1}{kg}, \frac{1}{c} = \frac{K}{J}, \text{ and } \frac{1}{B} = \frac{s}{1}$$

Combining the units for all the parameters, again, we get the units for k:

$$k = \left(\frac{J}{s * m^2 * K}\right)\left(\frac{m^2}{1}\right)\left(\frac{1}{kg}\right)\left(\frac{K}{J}\right)\left(\frac{s}{1}\right) = \frac{1}{kg}$$

### Task 1 Analysis:

The equation has now been rewritten with new variables defined in terms of the original parameters. This simplifies our solution code and reduce the risk of error.

## Task 2:

### Introduction:

The goal of Task 2 is to simplify the equation derived in Task 1 so that it's easier to analyze mathematically

### Task 2:

Modifying the equation we finished with in Task 1 so that it's easier to analyze mathematically, we introduce a new dependent and independent variable set:

$$T = 1 + \tilde{y}$$

Now we can alter the equation with this new definition where we assume that  $\tilde{y}$  is small:

$$\Rightarrow \frac{d\tilde{y}}{dt} = De^{-\frac{\gamma}{1+\tilde{y}}} - k\tilde{y}, \tilde{y}(0) = 0$$

Since  $\tilde{y}$  is a small number, we can prove that:

$$\frac{1}{1+\tilde{y}} \approx 1 - \tilde{y}$$

in order to further simplify our equation. As  $\tilde{y}$  approaches 0 from the positive end, we can see that both  $\frac{1}{1+\tilde{y}}$  and  $1 - \tilde{y}$  approach the value of 1 from the left side of 1 and are close to the same value. To show this, let's set  $\tilde{y}$  to some very small value, say 0.001 and solve for both equations:

$$\frac{1}{1+0.001} = 0.999001 \text{ and } 1 - 0.001 = 0.999000$$

So we can see that when  $\tilde{y}$  is very small, these two equations are basically equal to each other. In this scenario when  $\tilde{y} = 0.001$ , we see that the difference in values between the two equations was 0.000001 which is a very negligible difference, so in our equation we will let this equivalency be true and substitute it into the equation so that we get:

$$e^{-\frac{\gamma}{1+\tilde{y}}} \approx e^{-\gamma(1-\tilde{y})} = e^{-\gamma} e^{\gamma\tilde{y}}$$

So the original equation can be rewritten as:

$$\frac{d\tilde{y}}{dt} = De^{-\gamma} e^{\gamma\tilde{y}} - k\tilde{y}, \tilde{y}(0) = 0$$

Next let's define:

$$y = \gamma\tilde{y} \Rightarrow \tilde{y} = \frac{y}{\gamma}$$

and then substitute this variable into our equation:

$$\frac{1}{\gamma} \frac{dy}{dt} = De^{-\gamma} e^y - k\frac{y}{\gamma}, y(0) = 0$$

which transforms into:

$$\frac{dy}{dt} = \gamma De^{-\gamma} e^y - ky, y(0) = 0$$

From here, we can group some variables into one variable like we did before in Task 1. Let's define this variable as  $\tilde{D}$  and its equivalency is shown here:

$$\tilde{D} = \gamma D e^{-\gamma}$$

Substituting this into the equation we get:

$$\frac{dy}{dt} = \tilde{D} e^y - ky, \quad y(0) = 0$$

Now let's define:

$$t = \tilde{D} \tilde{t}$$

and then, using the chain rule with the differential equation  $\frac{dy}{dt}$ , we can find an equivalency for the equation:

$$\frac{dy}{dt} = \frac{dy}{d\tilde{t}} * \frac{d\tilde{t}}{dt}$$

We can see from our definition of  $\tilde{t}$  what  $\frac{d\tilde{t}}{dt}$  is:

$$\frac{d\tilde{t}}{dt} = \tilde{D}$$

so that we can substitute this into our chain rule equation:

$$\Rightarrow \frac{dy}{d\tilde{t}} = \frac{dy}{dt} * \tilde{D}$$

and then transform our ode to:

$$\frac{dy}{d\tilde{t}} * \tilde{D} = \tilde{D} e^y - ky \Rightarrow \frac{dy}{d\tilde{t}} = e^y - \frac{k}{\tilde{D}} y$$

We can group variables again into 1 variable here. Let's set this variable to be  $\alpha$ , and its equivalency is shown here:

$$\alpha = \frac{k}{\tilde{D}}$$

Finally, we transform everything into the last equation, which is an accumulation of all prior definitions of variables:

$$\text{was } \frac{dy}{d\tilde{t}} = e^y - \alpha y, \quad y(0) = 0$$

Where define our new variables in terms of the original parameters to be:

$$\alpha = \frac{k}{\tilde{D}} = \frac{k}{\gamma D e^{-\gamma}}$$

$$\tilde{D} = \gamma D e^{-\gamma}$$

## Task 2 Analysis:

Through the use of equivalency assumptions and redefinition of variables, we have even further simplified our equation from Task 1 into a form that is easier to analyze mathematically.

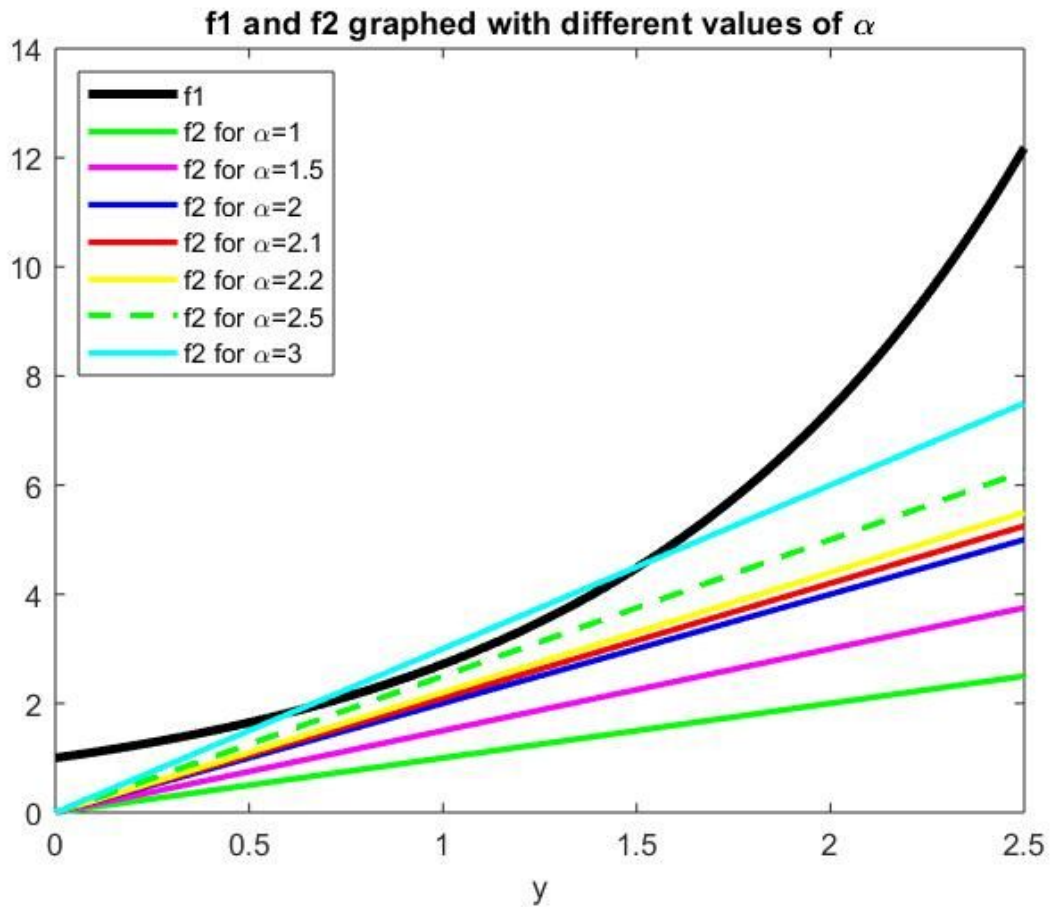
### Task 3:

#### Introduction:

The goal of Task 3 is to determine the critical points of the thermodynamic system with respect to  $t$ . By finding the critical points of the model, we are able to predict the behavior of the system as  $t$  approaches infinity and the system reaches an equilibrium state. The equations for  $f_1(y)$  and  $f_2(y)$  are:

$$f_1(y)=e^y \text{ and } f_2(y)=\alpha y$$

#### Task 3 Results:



**Figure 1:** The graphs of  $f_1(y)$  and  $f_2(y)$ , with  $f_2(y)$  graphed for the values of  $\alpha$  seen in the graph



### Task 3 Analysis:

Figure 1 shows that when  $\alpha=3$ , there are 2 critical points and when  $\alpha=2.5$  the graphs of  $f_1(y)$  and  $f_2(y)$  come very close but don't quite intersect. With a little more testing and estimating it can be seen that critical points emerge at approximately  $\alpha=2.7$ . At this value of  $\alpha$  there is only one critical point because this is the only intersection of the graphs of  $f_1$  and  $f_2$ . With a little manipulation of the equations  $f_1(y) = f_2(y)$ ,  $f'_1(y) = f'_2(y)$ , we find explicitly what  $\alpha$  and  $y$  equal at this initial intersection:

$$f_1(y) = f_2(y) \Rightarrow e^y = \alpha y \Rightarrow y = \ln |\alpha y|$$

$$f'_1(y) = f'_2(y) \Rightarrow e^y = \alpha \Rightarrow y = \ln |\alpha|$$

In order for both of these statements to be true, which they are,  $y$  must equal 1 and  $\alpha_*$  must equal  $e$ , now we can set our two variables  $\alpha_*$  and  $y_*$  to be  $\alpha_*=e$  and  $y_*=1$ . This point can be seen to be semi-stable when we utilize the derivative test on our given function  $\frac{dy}{dt} = e^y - \alpha y$ , and this process is shown below:

$$\text{Let } f(y) = e^y - \alpha y$$

$$f'(y) = e^y - \alpha$$

Plugging in  $y_*$  and  $\alpha_*$  we get:

$$f'(1) = e^1 - e = 0$$

The derivative test fails at this point because the derivative of our function at the critical point of our set  $\alpha$  is equal to 0, meaning that this point  $(y_*, \alpha_*)$  is semi-stable. When testing a  $y$  value below and above the critical point, we can see the stability above and below the point. Above the critical point we get:

$$f(1.1) = e^{1.1} - e(1.1) > 0$$

which indicates that the slope of the function is positive above the critical point, meaning that the function increases towards infinity rather than decreasing down to the critical point. Likewise below the critical point yields the result:

$$f(0.9) = e^{0.9} - e(0.9) > 0$$

which indicates that the slope of the function is also positive below the critical point, meaning that the function increases to the critical point where it will plateau. This creates the bifurcation point  $\alpha_*$ , where once you go above this value of  $\alpha$  there will be two critical points with the lower  $y$ -valued critical point being stable and the higher  $y$ -valued one being unstable. This splitting is called bifurcating and results in having a stable and unstable critical point for every value of  $\alpha > \alpha_*$ . This can be shown by evaluating the critical points of  $\alpha=3$ . The critical points of  $\alpha$  can be found by setting  $f_1(y)=f_2(y)$  to which we get the  $y$  values  $y=.61906$ , and  $1.5121$ . These points can be analyzed for stability by doing the derivative test to which will provide that  $f'_1(.61906)=-1.1428$  and  $f'_1(1.5121)=1.5362$  making the lower critical point stable and the

greater critical point unstable. Below  $\alpha_*$  there are no critical points. Critical points must always be a whole number, hence why our values for our critical points go from 0 below  $\alpha_*$  to 1 at  $\alpha_*$  and 2 at any value larger than  $\alpha_*$ . To form a bifurcation point as we did in this task there must always be a critical point where the derivative test doesn't succeed in determining whether the point is stable or unstable; we have this is the form of  $\alpha_*$ . It should be noted however that this is not the only time where the derivative test can fail.

## Task 4:

### Introduction:

Task 4 seeks to illustrate the behavior of the solution to equation 7 when  $\alpha < \alpha_*$ . From the equation know that when  $\alpha=0$  that the solution to the equation is simply

$$y = -\ln(1 - t)$$

### Task 4 Results and Analysis:

When  $\alpha < \alpha_*$  we would expect the solution to have no critical points. The graph of  $y = -\ln(1 - t)$  is essentially an exponential graph for  $t < 1$  and then the graph runs away to infinity. We can also see in Figure 1 in Task 3 that when  $\alpha < \alpha_*$  there is no critical point as  $e^y$  never equals  $\alpha y$  and there is no intersecting for  $\alpha$ 's less than  $\alpha_*$ .

## Task 5:

### Introduction:

The goal of Task 5 is to compare the solve accuracy of fixed vs. adaptive improved forward euler solvers. The first step is creating a function *yprime.m* that defines the right hand side of the model of the chemical equation. The second piece of code: *fixed.m* outlines the code required to solve the system using a fixed time step and the improved Euler method. The third and final piece of code for this task, *adaptive.m*, defines an improved Euler solver that uses an adaptive time step to improve the solve accuracy. We will then compare the accuracy of the two solvers to ensure that each exhibits the expected behavior at the critical points found in the previous task.

### Task 5 Results:

**Table 1:** The following values of *epsilon* were run in *adaptive.m*, the computational effort (*count*) and the computed value of  $y(0.999)$  are recorded in the table below:

Parameters for <i>adaptive.m</i>				
Epsilon:	Count	Computed $y(tfin)$	$y(tfin)$ exact	% error
.0005	536	6.8607	6.90776	1.4802
.001	384	6.8130	6.90776	1.3909
.003	244	6.6603	6.90776	3.7154

**Table 2:** The following values of *epsilon* were run in *adaptive.m*, the computational effort (*count*) and the computed value of  $y(0.999)$  are recorded in the table below:

Parameters for <i>fixed.m</i>				
Timestep (h)	Count	Computed $y(tfin)$	$y(tfin)$ exact	% error
.001	1998	6.8409	6.90776	0.9774
.002	1000	6.7550	6.90776	2.2614
.003	666	6.5811	6.90776	4.9636

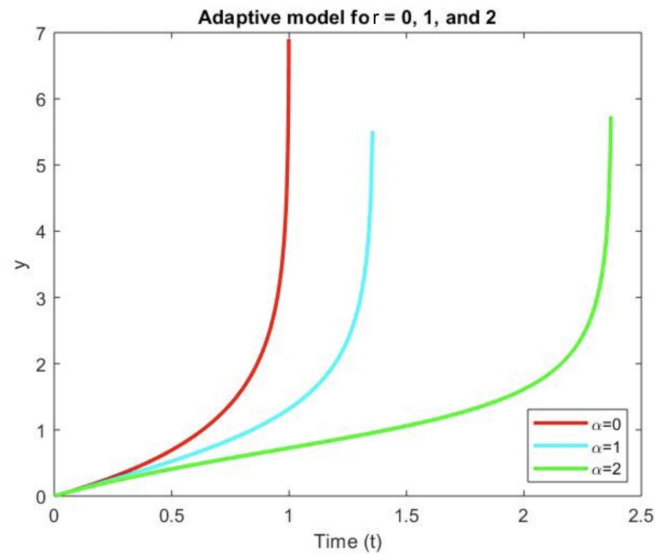
### Task 5 Analysis:

Relative to *adaptive.m* simulations run with *fixed.m* were more computationally expensive as indicated by the larger values of *count*. *Adaptive.m* required a count of 536 to compute  $y(tfin)$  with 1.48% error. From the tests, one can see that it would take a *count* somewhere between 1000 and 1998 with *h* between 0.002 and 0.001 using *fixed.m* to achieve a similar level of precision to *adaptive.m* with *epsilon* set at 0.0005. It could be argued that the use *fixed.m* to find the solution to this equation would be a poor choice as *adaptive.m* can provide a better solution with a lower count.

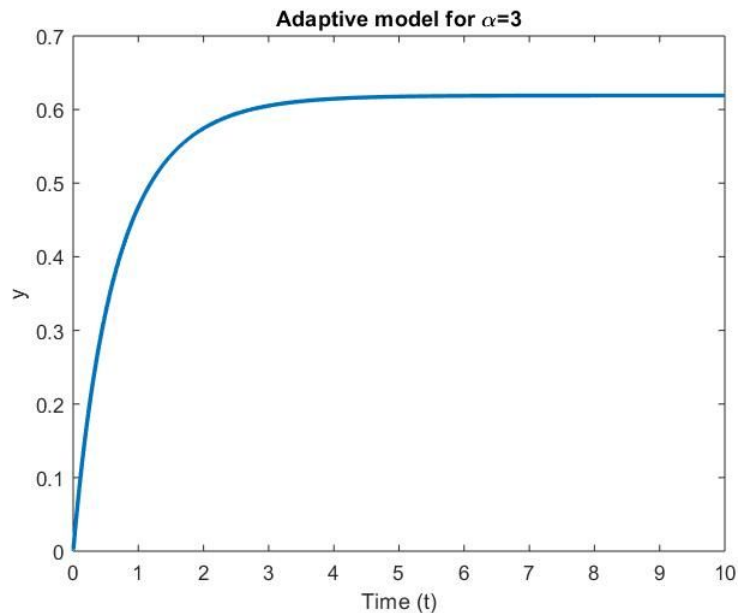
## Task 6:

### Introduction:

The goal of Task 6 was to analyze how changing the value of  $\alpha$  affects the graph of  $y$  vs  $t$  in the adaptive model and then once the value of  $\alpha$  is greater than  $\alpha_*$  (when  $\alpha=3$ ), how that will affect the graph of  $y$  vs  $t$ .



**Figure 2:** This is the plot for the adaptive improved Euler method with  $\alpha=0, 1, \text{ and } 2$



**Figure 3:** This is the plot for the adaptive improved Euler method with  $\alpha=3$

**Task 6 Results and Analysis:**

As the value of  $\alpha$  increases in figure 2, the time that it takes for the slope of the graph to become nearly vertical increases and each of these graphs represents a similar shape to that of the graph of  $y = -\ln(1-t)$  in Task 4. It takes a longer and longer time because the value of  $\alpha$  forces it to. Then comes figure 3 where an  $\alpha$  value of 3 is used and this graph looks entirely different. It immediately as an exponential spike to about  $y = .6$  and then levels off. This is due to there being a stable critical point at  $y = .61906$ . All the graphs of any  $\alpha$  value less than  $\alpha = e$  will not have an asymptote as there is no critical point for those values of  $\alpha$ . It can also be tested to prove that  $y = .61906$  is a stable critical point as shown in Task 3 by setting the  $y_{init} > .619106$  as it will then go down an asymptote at the same value. This proves what was found in Task 3 as every value of  $\alpha < e$  has no critical point and thus is infinitely vertical whereas any value of  $\alpha > e$  will have a horizontal asymptote at the stable critical point.

## Task 7:

### Introduction:

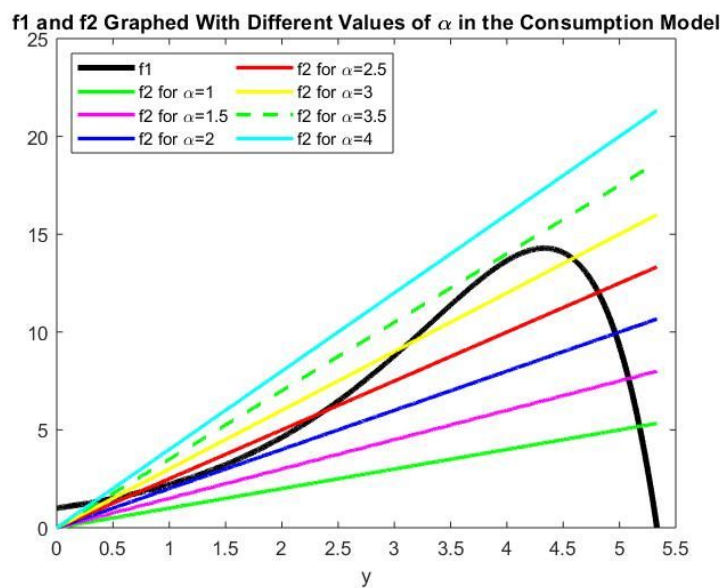
The goal of Task 7 is to model the consumption of combustible material that is the fuel source of the reaction. A simplified model is given by modifying the previous equation used to model the reaction. The new equation that we will consider is:

$$dy/dt = (1 - \beta y)e^y - \alpha y, y(0) = 0$$

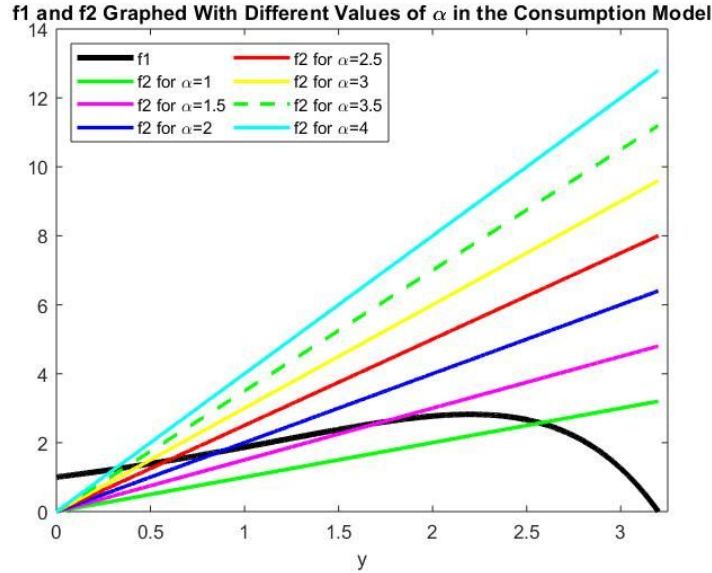
For the figures in this Task we split this equation into two different equations:

$$f_1(y) = (1 - \beta y)e^y$$

$$f_2(y) = \alpha y$$



**Figure 4:**  $f_1(y)$  and  $f_2(y)$  graphed for the consumption model with varying values of  $\alpha$  with  $\beta=3/16$



**Figure 5:**  $f_1(y)$  and  $f_2(y)$  graphed for the consumption model with varying values of  $\alpha$  with  $\beta=5/16$

### Task 7 Analysis:

Equation 12 models the combustion equation previously considered by introducing the dependence of the equation on finite fuel. By introducing into the equation  $(1 - \beta y)$  it shows that as  $y$  increases, if the value of  $\beta > 0$  then  $f_1(y)$  will eventually be equal to 0 whereas  $f_1(y)$  in Task 3 would exponentially increase which doesn't model consumption at all as the fuel is infinite in that case. So by adding  $(1 - \beta y)$  this provides a finite fuel source that is consumed as  $y$  increases. When  $y = 1/\beta$  the rate of combustion is now  $(1 - \beta(1/\beta))e^y - \alpha y$ . We can see that  $\beta(1/\beta) = 1$  and  $1 - 1 = 0$  so when  $y = 1/\beta$ , the equation is now  $dy/dt = 0 \cdot e^y - \alpha y = -\alpha y$ . At this point all of the fuel must be gone and the rate of combustion is not increasing. The heat generating term in the equation is  $(1 - \beta y)e^y$  and at the point where  $y = 1/\beta$  this term is 0, meaning no heat is being generated due to the loss of a fuel source.

1. There is a critical point for every  $\alpha > 0$  when  $\beta > 0$  because as seen in figures 4 and 5, when  $\alpha$  is large  $f_2(y)$  will intersect  $f_1(y)$  when  $y < 1$  and when  $\alpha$  is small, the plot of  $f_1$  will always go to  $y=0$  as explained above, so  $f_2(y)$  will then intersect  $f_1(y)$  around when  $f_1(y)$  is nearing  $y=0$  as seen in both figures 4 and 5 when  $\alpha=1$ .

In figures 4 and 5 it can be seen that immediately before  $f_1(y)$  and  $f_2(y)$  at the largest value of  $y$  for each  $\alpha$  that  $f_1(y) > f_2(y)$ . This implies that the slope of  $y$  is going from positive to negative, which according to the phase diagrams that were discussed in class, means this critical point is going to be stable. This occurs for every value of  $\alpha$  at the last intersection of  $f_1(y)$  and  $f_2(y)$ .



2. At the bifurcation points that occur in the graph,  $f_1(y)$  and  $f_2(y)$  are always tangential to one another. This implies that  $f_1(y)=f_2(y)$  as well as  $f'_1(y) = f'_2(y)$ , where  $f_1(y)=(1-\square y)e^y$  <sup>[1]</sup>,  $f_2(y)=\alpha y$  <sup>[2]</sup>,  $f'_1(y)=e^y(1-\square-\square y)$  <sup>[3]</sup>, and  $f'_2(y)=\alpha$  <sup>[4]</sup>. This gives us  $(1-\square y)e^y=\alpha y$  <sup>[5]</sup> and  $e^y(1-\square-\square y)=\alpha$  <sup>[6]</sup>. Solving [5] for  $e^y$  shows that  $e^y=(\alpha y)/(1-\square y)$  <sup>[7]</sup>. By plugging [7] into [6] it is found that  $(\alpha y)/(1-\square y) (1-\square-\square y)=\alpha$  which can be simplified to the equation  $\square y^2-y+1=0$  <sup>[8]</sup>. Using the quadratic formula where  $b=-1$ ,  $a=\beta$ , and  $c=1$  we get  $y = \frac{1 \pm \sqrt{(-1)^2 - 4(\beta)(1)}}{2\beta} = y = \frac{1 \pm \sqrt{1-4\beta}}{2\beta}$  <sup>[9]</sup>. From [9] we see that a bifurcation point exists when  $1-4\beta=0$  which implies that  $\beta_*=1/4$ .

From [9] we get the equations  $y_1 = \frac{1-\sqrt{1-4\beta}}{2\beta}$  <sup>[10]</sup> and  $y_2 = \frac{1+\sqrt{1-4\beta}}{2\beta}$  <sup>[11]</sup> which will then be plugged into [6] to find the values of  $\alpha_1$  and  $\alpha_2$ . This gives us  $\alpha_1 = e^{\frac{1-\sqrt{1-4\beta}}{2\beta}} (1-\beta - \frac{1-\sqrt{1-4\beta}}{2})$  <sup>[12]</sup> and  $\alpha_2 = e^{\frac{1+\sqrt{1-4\beta}}{2\beta}} (1-\beta - \frac{1+\sqrt{1-4\beta}}{2})$  <sup>[13]</sup> where  $\beta y_1 = \frac{1-\sqrt{1-4\beta}}{2}$  and  $\beta y_2 = \frac{1+\sqrt{1-4\beta}}{2}$ . This proves that  $\beta y_1 > 0$  and  $\beta y_2 < 1$  because  $\frac{1-\sqrt{1-4\beta}}{2} > 0 \rightarrow 1 - \sqrt{1-4\beta} > 0 \rightarrow 1 > \sqrt{1-4\beta}$  which is true for all values of  $0 < \beta < \frac{1}{4}$  meaning  $\beta y_1 > 0$  and  $\frac{1+\sqrt{1-4\beta}}{2} < 1 \rightarrow 1 + \sqrt{1-4\beta} < 2 \rightarrow \sqrt{1-4\beta} < 1$  which once again, is true for all values  $0 < \beta < \frac{1}{4}$  meaning  $\beta y_2 < 1$ .

3. When  $\beta < \beta_*$  (aka  $1/4$ ) and  $\alpha_1 < \alpha < \alpha_2$  there are three critical points. By viewing the figure 4 it can be seen that multiple values of  $\alpha$  have 3 intersecting points (and  $\beta=3/16$  is indeed less than  $1/4$ ) so it is safe to assume that this figure accurately represents the plot of this at least in general shape. The three critical points will, in order from smallest to largest, be stable at the first, unstable at the second, and stable at the third. For the largest critical point we showed in Task 7.1 that this will always be stable and for the same reasoning, the smallest critical point is also stable because  $f_1(y) > f_2(y)$ . This same reasoning except used in the case where  $f_2(y) > f_1(y)$  shows that the second critical point will always be unstable as the slope of  $y$  at this point goes from negative to positive, producing an unstable critical point. Setting an initial condition of  $y(0)=y_0$  where  $\beta y_0$  is near 1 would show that the smallest  $y_{init}$  could be is  $y_0 > 4$  because  $\beta < \beta_*$  and  $\beta_*=1/4$ . This would mean that based on the values of  $\alpha$  plotted in figure 4, that the most critical points any value of  $\alpha$  can have is 1 because this is after the plot of  $f_1(y)$  begins to slope down.
- 4.
- Similarities for (i)
    - There are no critical points for small values of  $y$  in either case
  - Differences for (i)

- i. There are critical points for these small values of  $\alpha$  in the adjusted model where in the original there were none
    - ii. The temp for the original model goes to infinity while the temperature for the adjusted model goes to 0
  - c. Similarities for (ii)
    - i. Both contain a bifurcation point
    - ii. Both contain critical points
  - d. Differences for (ii)
    - i. The original model contains either 1 or 2 critical points for any  $\alpha$
    - ii. The adjusted model contains either 2 or 3 critical points for any given  $\alpha$
  - e. Similarities for (iii)
    - i. Both reach a critical point very quickly that is stable
  - f. Differences for (iii)
    - i. The original model contains 2 critical points for all  $\alpha$ , the first of which is stable and the second is unstable
    - ii. The adjusted model only contains 1 critical point which is always stable
5. Substituting  $\beta=3/16$  into  $\alpha_1, \alpha_2, y_1$ , and  $y_2$  we get that

$$y_1 = \frac{1 - \sqrt{1 - 4(3/16)}}{2(3/16)} = 4/3,$$

$$y_2 = \frac{1 + \sqrt{1 - 4(3/16)}}{2(3/16)} = 4,$$

$$\alpha_1 = e^{\frac{1 - \sqrt{1 - 4(3/16)}}{2(3/16)}} \left( 1 - 3/16 - \frac{1 - \sqrt{1 - 4(3/16)}}{2} \right) = 2.1339, \text{ and lastly}$$

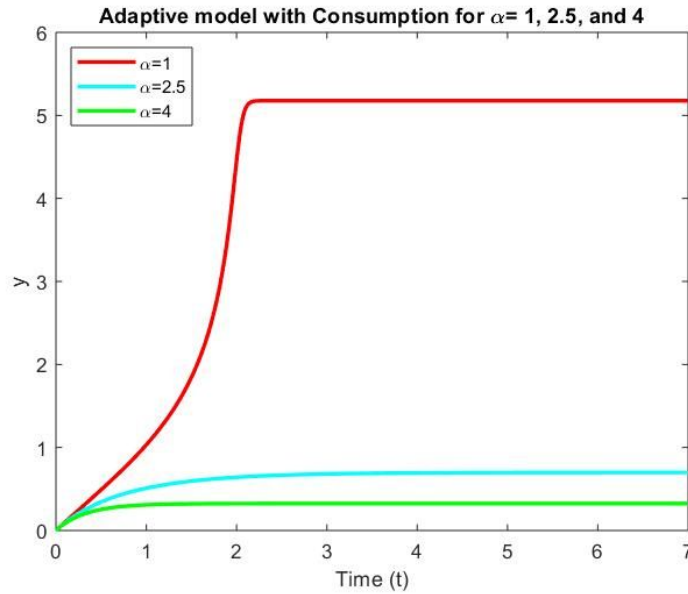
$$\alpha_2 = e^{\frac{1 + \sqrt{1 - 4(3/16)}}{2(3/16)}} \left( 1 - 3/16 - \frac{1 + \sqrt{1 - 4(3/16)}}{2} \right) = 3.4124$$

- a. The three categories of  $\alpha$ 
    - i.  $\alpha < \alpha_1$ 
      - 1.  $\alpha=1, 1.5, 2$
    - ii.  $\alpha_1 < \alpha < \alpha_2$ 
      - 1.  $\alpha=2.5, 3$
    - iii.  $\alpha_2 < \alpha$ 
      - 1.  $\alpha=4$
6. Off the bat, the value of  $\beta$  is greater than  $\beta_*$  which means that  $\sqrt{1 - 4\beta}$  has to exist as a real number ( $1 - 4\beta > 0$ ) in order for a bifurcation point to exist. With this knowledge it can safely be stated that there are no bifurcation point in figure 5 which is reflected in the figure as every value of  $\alpha$  has only one critical point. This implies that there is not a bifurcation point as for that to happen there would need to be a value of  $\alpha$  (is deemed  $\alpha_*$ ) where there is only one critical point that is unstable, and any  $\alpha < \alpha_*$  has no critical points and every  $\alpha > \alpha_*$  has 2 critical points with the lower one being stable and the higher one being unstable.

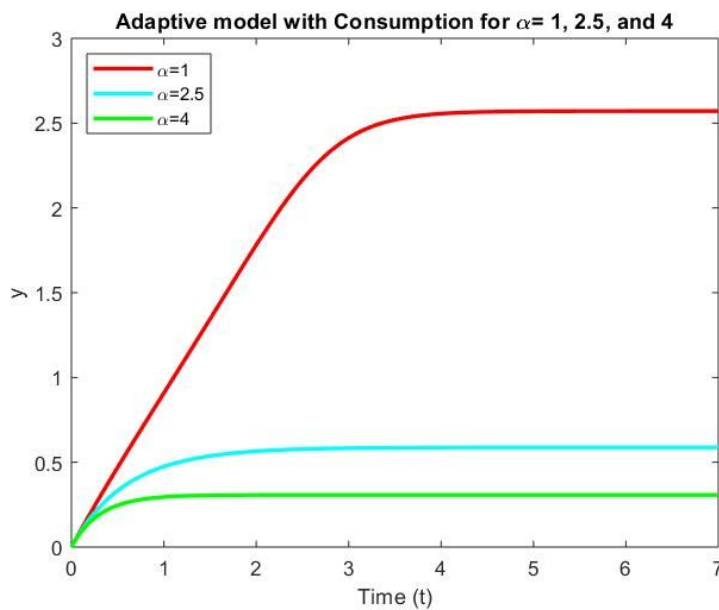
## Task 8:

### Introduction:

The goal of Task 8 is to implement the adaptive improved Euler program and apply the new given function (12) to the model of the situation. With this and the added  $(1-\beta y)$  to the equation we can create a graph using our adaptive model made in Task 5. Using this we can compare and see how the new solution including  $\beta$  affects the solution.



**Figure 6:** Plot for adaptive improve Euler with varying values of  $\alpha$ ,  $\beta=3/16$ , and  $t_{fin}=7$



**Figure 7:** Plot for adaptive improve Euler with varying values of  $\alpha$ ,  $\beta=5/16$ , and  $t_{fin}=7$

### Task 8 Analysis:

In figure 6, the solutions at  $\alpha=4$  and  $\alpha=2.5$  behave similarly where the slope constantly decreases until the function plateaus at a certain value, but the solution at  $\alpha=1$  behaves differently where the model exponentially increases until plateauing at a value. Comparing the new function with what was found using equation (7) at  $\alpha=1$ , you can see that, in figure 6, the segment of the function from  $0 \leq t \leq 2$  follows a similar pathway that the function in figure 2 has where it's exponentially increasing until it's almost a vertical line, the difference is that in figure 6 the function plateaus after this increase at around  $t = 2$  whereas the function never plateaus in figure 2. The results for each value of  $\alpha$ , shown in figure 6, and consistent with the intersections found in Task 7, shown in figure 4, because these intersections show where the critical points occur since when  $f_1=f_2$  the differential equation (12) is equal to 0. These critical points (discovered by the intersections of figure 4 in Task 7) are then translated to the points where the function will plateau to when evaluated, which is exactly what was depicted when we used *adaptive.m* to solve for equation (12) and graphed it in figure 6. The lines plateau at those values because those are the stable critical point values for the solution with their given  $\alpha$ 's

When looking at figure 7 which is made with a  $\beta = \frac{5}{16}$ , we can see that the solutions for  $\alpha = 2.5$  and  $\alpha = 4$  are similar to those in figure 6. They follow the same trends and at essentially the same values. When comparing the two figures for their plots with  $\alpha = 1$ , we see a completely different solution in figure 7. The line follows a more linear path until  $t = 2$  when it plateaus gently comparatively to the more harsh transition in figure 6. Looking at figure 5 and figure 7 we can see that the intersections of  $f_1$  and  $f_2$  for  $\alpha = 1$  are a little above 2.5 and when looking at  $\alpha = 1$  in figure 7 we can see that it plateaus as expected at right above 2.5. This is the case with  $\alpha = 2.5$  and 4 for these graphs.

For the value of  $\alpha = 1$  and  $\beta = \frac{5}{16}$ , we see that the graph actually looks linear from  $0 \leq t \leq 2$ . This is because when looking at figure 5, we can see that the  $f_1$  plot is almost in line with the  $f_2$  with  $\alpha = 1$ . At this point,  $\frac{dy}{dt}$  is a really small number so the slope isn't changing very much at all.  $\alpha = 1$  is the only  $\alpha$  value that has a situation like this and the other  $\alpha$  values. In addition to this, as talked about in Task 7.6, there is no bifurcation point when  $\beta = \frac{5}{16}$  and thus leaving  $\beta = \frac{3}{16}$  as a much better model for the equation.

## Conclusion & Summary

1. In Task 1, a very complex equation that was dependent on lots of parameters was nondimensionalized. By defining extraneous parameters in terms of other fundamental parameters, the chance of producing error by incorrectly defining a parameter reduced to only a few parameters. This also makes putting the equation into MATLAB easier and less coding is required.
2. In Task 2, the nondimensionalized equation from Task 1 was simplified even more by assuming equivalency of some parameters in the equation. This will also make it easier to program in MATLAB.
3. In Task three, each of the critical points for the solution was determined graphically for different values of  $\alpha$ . Multiple values of alpha were tested until points where the critical behavior was found. In this section the bifurcation behavior was analyzed with along with the derivative behavior of  $f_1$  to determine the stability of the critical points and the presence of a bifurcation when the derivative test failed. In addition, the value of  $\alpha_*$  was determined. In addition, it was determined that for every  $\alpha > \alpha_*$ , there was a stable and unstable critical point.
4. Task 4 focused on analyzing the solution space behavior for  $\alpha < \alpha_*$ . Rigorous analysis revealed that when  $\alpha$  is less than  $\alpha_*$  the solution would not have any critical points. This fact is implied by the lack of intersections of  $f_1$  by function solutions with  $\alpha < e$ .
5. The computational efficiency of adaptive versus fixed time step solving was analyzed in Task 5. It became clear that an adaptive method was far more computationally efficient than a fixed time step solver to achieve a desired error tolerance. An adaptive scheme was capable of producing a solution with a similar amount of error as fixed width in roughly half the number of computation or *count*.
6. In Task 6, the effects of  $\alpha$  on the time it takes for  $y(t)$  to reach the critical point was compared. From this study, one sees that as the value of  $\alpha$  increases, the initial rate of change decreases and the solution takes longer to run away. This is obvious because the  $\alpha$  term in the equation subtracts from the other term, decreasing  $dy/dt$ . On the other hand, when  $\alpha$  is greater than  $\alpha_*$ , one can see that the solution quickly reaches equilibrium in Figure 3
7. This Task was the most complicated analytically. Task 7 began by introducing a new equation that included that modeled the amount of fuel available for combustion. We then analyzed different values of  $\beta$  and their impact on the behavior of the solution. Our analysis shows that the  $\beta$  factor places an additional constraint on the rate of change, ultimately providing a physical maximum to the rate of combustion.
8. Task 8 implemented the adaptive improved Euler on the new equation introduced with the consumption model. In this Task, different values of  $\alpha$  and  $\beta$  were tested in order to observe the critical behavior of different parameter combinations. We found that the analysis of the critical points we found in earlier Tasks was verified with these models.

## Appendix 1: Task 3

### (Figure 1)

```
%setting the given values and creating arrays of the right size.
y1=0:.00001:2.5;
f1=zeros(length(y1),1);
f21=zeros(length(y1),1);
f22=zeros(length(y1),1);
f23=zeros(length(y1),1);
f24=zeros(length(y1),1);
f25=zeros(length(y1),1);
f26=zeros(length(y1),1);
f27=zeros(length(y1),1);
%creating the array for f1
for ii=1:length(y1)
    f1(ii)=exp(y1(ii));
end
%creating the solutions for the different values of alpha
for ii=1:length(y1)
    alpha=1;
    f21(ii)=(alpha)*y1(ii);
end
for ii=1:length(y1)
    alpha=1.5;
    f22(ii)=(alpha)*y1(ii);
end
for ii=1:length(y1)
    alpha=2;
    f23(ii)=(alpha)*y1(ii);
end
for ii=1:length(y1)
    alpha=2.1;
    f24(ii)=(alpha)*y1(ii);
end
for ii=1:length(y1)
    alpha=2.2;
    f25(ii)=(alpha)*y1(ii);
end
for ii=1:length(y1)
    alpha=2.5;
    f26(ii)=(alpha)*y1(ii);
end
for ii=1:length(y1)
    alpha=3;
    f27(ii)=(alpha)*y1(ii);
end
%Plotting the data.
figure(1)
plot(y1,f1,'k','LineWidth',3.0)
hold on
plot(y1,f21,'g',y1,f22,'m',y1,f23,'b',y1,f24,'r',y1,f25,'y',y1,f26,'g--',y1,f27,
'c','LineWidth',2.0)
```

```
xlabel('y');  
title('f1 and f2 graphed with different values of \alpha')  
legend('f1','f2 for \alpha=1', 'f2 for \alpha=1.5', 'f2 for \alpha=2','f2 for  
\alpha=2.1','f2 for \alpha=2.2','f2 for \alpha=2.5','f2 for  
\alpha=3','Location','nw')
```

## Appendix A: Task 5

### yprime.m

```
%Creating function for right hand side of y-prime.
function f=yprime(t,y)
global alpha
global count
f= (exp(y))-alpha*y;
count=count+1;
end
```

### Adaptive.m

```
%calling and setting global variables
global count
count=0;
global alpha
%calling inputs from Adaptive.dat and printing them
[yinit, tinit,tfin, h, alpha, epsilon, hraise,
hcut,hbig,hsmall]=textread('adaptive.dat','%f %f %f %f %f %f %f %f %f')
%creating arrays and setting initial conditions
y=[];
t=[];
%Initializing variables for the loop
j=1;
y(j)=yinit;
t(j)=tinit;
nsteps=0;
%adaptive improved Euler while loop
while (t<tfin)
    tnew=t(j)+h;
    %checking that if tnew is greater than tfin, it will set tnew=tfin to run the
    %loop one more time
    if tnew>tfin
        tnew=tfin;
        h=tnew-t(j);
    End
    %creating the values k0,ylow,k1, and yhigh that will be used to check the
    %error within this equation to make sure it is less than epsilon
    k0=yprime(t(j),y(j));
    ylow=y(j) + h*k0;
    k1=yprime(tnew,ylow);
    yhigh=y(j)+((h/2)*(k0+k1));
    %checking that the error is less than epsilon and if it is, then raising h by
    %a factor of hraise, adding 1 to j and then yetting y(j) to yhigh and t(j) to
    %tnew in order to set the intial values for the next loop
    if abs((ylow-yhigh)/abs(yhigh))<epsilon
        h=h*hraise;
        j=j+1;
        y(j)=yhigh;
        t(j)=tnew;
    %If h ends up bigger than hbig when we raise it by hraise, we will then set it
    %equal to hbig
    if h>hbig
```



```

        h=hbig;
    end
    %If the error was not less than epsilon we divide h by hcut to then run the
    next loop with a small h
    else
        h=h/hcut;
    %If h is too small then we output an error message that its too small
    if h<hsmall
        error('Too small of a time step')
    end
    end
    %add 1 to the value of nsteps
    nsteps=nsteps+1;
end
%the loop will continue going until t=tfin at which point it will finish and
then print out the following values
%outputting resulting variables
count
nsteps
yhigh
tnew

%for epsilon=.0005, count= 536 and yhigh=6.8607
%for epsilon=.003 count=244 yhigh=6.6603
%for epsilon=.001 count=384 and yhigh=6.8130

```

### Adaptive.dat

```
0 0 .999 1 0 0.0005 1.05 2.0 1 1.0000e-12
```

### Fixed.m

```

%initializing global variables and creating the proper arrays.
global alpha
global count
[yinit, tinit, tfin, h, alpha]=textread('fixed.dat','%f %f %f %f %f');
y=[];
t=[];
j=1;
%Setting given values
count=0;
y(j)=yinit;
t(j)=tinit;
nsteps=0;
%Creating solution using a fixed euler method
while (t(j)<tfin)
    j=j+1;
    t(j)=t(j-1)+h;
    if t(j)>tfin
        t(j)=tfin;
        h=t(j)-t(j-1);
    end
    k0=yprime(t(j-1),y(j-1));
    ypred=y(j-1)+h*(k0);

```

```

        k1=yprime(t(j),ypred);
        y(j)=y(j-1)+[(h/2)*(k0+k1)];
        nsteps=nsteps+1;
end

%outputting resulting variables
count
nsteps
ynew=y(j);
ynew
tnew=t(j);
tnew

%for h=.001 count=1998 and ynew=6.8409
%for h=.002 count=1000 and ynew=6.7550
%for h=.003 count=666 and ynew=6.5811

```

### **Fixed.dat**

```
0 0 0.999 .003 0
```

## Appendix 2: Task 6

### yprime.m

```
%Creating function for right hand side of y-prime.
function f=yprime(t,y)
global alpha
global count
f= (exp(y))-alpha*y;
count=count+1;
end
```

### Adaptask6.m (Figure 2 and 3)

```
%Calling and setting global variables
global count
count=0;
global alpha
%outputting the resulting data
[yinit, tinit, h, error, hraise, hcut,hbig,hsmall]=textread('adaptive.dat','%f
%f %f %f %f %f %f');
%creating arrays and setting initial values.
y1=[];
t1=[];
j=1;
y1(j)=yinit;
t1(j)=tinit;
nsteps=0;
alpha=0;
tfin1=.999;
%adaptive improved Euler while loop
while (t1<tfin1)
    tnew=t1(j)+h;
    if tnew>tfin1
        tnew=tfin1;
        h=tnew-t1(j);
    end
    k0=yprime(t1(j),y1(j));
    ylow=y1(j) + h*k0;
    k1=yprime(tnew,ylow);
    yhigh=y1(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y1(j)=yhigh;
        t1(j)=tnew;
        if h>hbig
            h=hbig;
        end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
end
```

```

        nsteps=nsteps+1;
end
%Plotting figure 2
figure(1)
plot(t1,y1,'r','LineWidth',2.0)
hold on
xlabel('Time (t)')
ylabel('y')
title('Adaptive model fo\alpha= 0, 1, and 2')

%Creating arrays and assigning initial condition
t2=[];
y2=[];
y2(j)=yinit;
t2(j)=tinit;
alpha=1;
tfin2=1.355;
%adaptive improved Euler while loop for alpha=1
while (t2<tfin2)
    tnew=t2(j)+h;
    if tnew>tfin2
        tnew=tfin2;
        h=tnew-t2(j);
    end
    k0=yprime(t2(j),y2(j));
    ylow=y2(j) + h*k0;
    k1=yprime(tnew,ylow);
    yhigh=y2(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y2(j)=yhigh;
        t2(j)=tnew;
    if h>hbig
        h=hbig;
    end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
    nsteps=nsteps+1;
end

plot(t2,y2,'c','LineWidth',2.0)
hold on

%creating arrays and setting initial values
t3=[];
y3=[];
t3(j)=tinit;
y3(j)=yinit;
alpha=2;

```

```

tfin3=2.37;
%adaptive improved Euler for alpha=2
while (t3<tfin3)
    tnew=t3(j)+h;
    if tnew>tfin3
        tnew=tfin3;
        h=tnew-t3(j);
    end
    k0=yprime(t3(j),y3(j));
    ylow=y3(j) + h*k0;
    k1=yprime(tnew,ylow);
    yhigh=y3(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y3(j)=yhigh;
        t3(j)=tnew;
    if h>hbig
        h=hbig;
    end
    else
        h=h/hcut;
    if h<hsmall
        error('Too small of a time step')
    end
    end
    nsteps=nsteps+1;
end

plot(t3,y3,'g','LineWidth',2.0)
legend('\alpha=0','\alpha=1','\alpha=2','Location','SE')
%creating arrays and setting initial values
t4=[];
y4=[];
t4(j)=tinit;
y4(j)=yinit;
alpha=3;
tfin4=10;
%adaptive improved Euler for alpha=3
while (t4<tfin4)
    tnew=t4(j)+h;
    if tnew>tfin4
        tnew=tfin4;
        h=tnew-t4(j);
    end
    k0=yprime(t4(j),y4(j));
    ylow=y4(j) + h*k0;
    k1=yprime(tnew,ylow);
    yhigh=y4(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y4(j)=yhigh;
        t4(j)=tnew;

```

```

        if h>hbig
            h=hbig;
        end
        else
            h=h/hcut;
            if h<hsmall
                error('Too small of a time step')
            end
        end
        nsteps=nsteps+1;
    end

    %Plotting figure 3
    figure(2)
    plot(t4,y4,'LineWidth',2.0)
    xlabel('Time (t)')
    ylabel('y')
    title('Adaptive model for \alpha=3')

    changed adaptive.dat for this section
    0 0 1 0.00005 1.05 2.0 1 1.0000e-12

```

## Appendix 3: Task 7

### Task 7 (Figure 4)

```
%creating arrays and setting initial values.
beta=(3/16);
y=0:.001:(1/beta);
f1=zeros(length(y),1);
f21=zeros(length(y),1);
f22=zeros(length(y),1);
f23=zeros(length(y),1);
f24=zeros(length(y),1);
f25=zeros(length(y),1);
f26=zeros(length(y),1);
f27=zeros(length(y),1);

%array for f1
for i=1:length(y)
    f1(i)=(1-(beta*y(i)))*exp(y(i));
end

%Creating solutions for the different values of alpha
for i=1:length(y)
    alpha=1;
    f21(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=1.5;
    f22(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=2;
    f23(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=2.5;
    f24(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=3;
    f25(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=3.5;
    f26(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=4;
    f27(i)=(alpha)*y(i);
end
%Creating and plotting the graph
figure(1)
plot(y,f1,'k','LineWidth',3.0)
hold on
```

```

plot(y,f21,'g',y,f22,'m',y,f23,'b',y,f24,'r',y,f25,'y',y,f26,'g--',y,f27,'c','L
ineWidth',2.0)
xlim([0 5.5])
xlabel('y');
title('f1 and f2 Graphed With Different Values of \alpha in the Consumption
Model')
legend({'f1','f2 for \alpha=1','f2 for \alpha=1.5','f2 for \alpha=2','f2 for
\alpha=2.5','f2 for \alpha=3','f2 for \alpha=3.5','f2 for
\alpha=4'},'Location','nw','NumColumns',2)

```

### Task 7 (Figure 5)

```

%creating arrays and setting initial values.
beta=(5/16);
y=0:.001:(1/beta);
f1=zeros(length(y),1);
f21=zeros(length(y),1);
f22=zeros(length(y),1);
f23=zeros(length(y),1);
f24=zeros(length(y),1);
f25=zeros(length(y),1);
f26=zeros(length(y),1);
f27=zeros(length(y),1);

%array for f1
for i=1:length(y)
    f1(i)=(1-(beta*y(i)))*exp(y(i));
End

%Creating solutions for the different values of alpha
for i=1:length(y)
    alpha=1;
    f21(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=1.5;
    f22(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=2;
    f23(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=2.5;
    f24(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=3;
    f25(i)=(alpha)*y(i);
end
for i=1:length(y)
    alpha=3.5;
    f26(i)=(alpha)*y(i);
end
end

```



```

for i=1:length(y)
    alpha=4;
    f27(i)=(alpha)*y(i);
end
%Plotting all of the solutions on a graph together with different colors.
figure(1)
plot(y,f1,'k','LineWidth',3.0)
hold on
plot(y,f21,'g',y,f22,'m',y,f23,'b',y,f24,'r',y,f25,'y',y,f26,'g--',y,f27,'c','L
ineWidth',2.0)
xlim([0 3.25])
xlabel('y');
title('f1 and f2 Graphed With Different Values of \alpha in the Consumption
Model')
legend({'f1','f2 for \alpha=1','f2 for \alpha=1.5','f2 for \alpha=2','f2 for
\alpha=2.5','f2 for \alpha=3','f2 for \alpha=3.5','f2 for
\alpha=4'},'Location','nw','NumColumns',2)

```

## Appendix B: Task 8

### adaptive.m (Figure 6)

```
%Assigning global variables
global count
count=0;
global alpha
global beta
beta=(3/16);
[yinit, tinit,tfin, h, error, hraise,
hcut,hbig,hsmall]=textread('adaptive.dat','%f %f %f %f %f %f %f %f');
%Creating arrays and assigning initial values for the alpha = 1 graph.
y1=[];
t1=[];
j=1;
y1(j)=yinit;
t1(j)=tinit;
nsteps=0;
alpha=1;
%Using the Adaptive Euler method for alpha = 1
while (t1<tfin)
    tnew=t1(j)+h;
    if tnew>tfin
        tnew=tfin;
        h=tnew-t1(j);
    end
    k0=yprime2(t1(j),y1(j));
    ylow=y1(j) + h*k0;
    k1=yprime2(tnew,ylow);
    yhigh=y1(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y1(j)=yhigh;
        t1(j)=tnew;
        if h>hbig
            h=hbig;
        end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
    nsteps=nsteps+1;
end
%creating the graph for all the plots to go on to
figure(1)
plot(t1,y1,'r','LineWidth',2.0)
hold on
xlabel('Time (t)')
ylabel('y')
title('Adaptive model with Consumption for \alpha= 1, 2.5, and 4')
```

```

%creating arrays and setting initial values for a new alpha value
t2=[];
y2=[];
y2(j)=yinit;
t2(j)=tinit;
alpha=2.5;
%Using the Adaptive Euler method for alpha = 2.5
while (t2<tfin)
    tnew=t2(j)+h;
    if tnew>tfin
        tnew=tfin;
        h=tnew-t2(j);
    end
    k0=yprime2(t2(j),y2(j));
    ylow=y2(j) + h*k0;
    k1=yprime2(tnew,ylow);
    yhigh=y2(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y2(j)=yhigh;
        t2(j)=tnew;
        if h>hbig
            h=hbig;
        end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
    nsteps=nsteps+1;
end
%Plotting solution with alpha = 2.5
plot(t2,y2,'c','LineWidth',2.0)
hold on

```

```

%creating arrays and setting initial values for a new alpha value
t3=[];
y3=[];
t3(j)=tinit;
y3(j)=yinit;
alpha=4;
%Using the Adaptive Euler method for alpha = 4
while (t3<tfin)
    tnew=t3(j)+h;
    if tnew>tfin
        tnew=tfin;
        h=tnew-t3(j);
    end
    k0=yprime2(t3(j),y3(j));
    ylow=y3(j) + h*k0;
    k1=yprime2(tnew,ylow);

```

```

yhigh=y3(j)+((h/2)*(k0+k1));
if abs((ylow-yhigh)/abs(yhigh))<error
    h=h*hraise;
    j=j+1;
    y3(j)=yhigh;
    t3(j)=tnew;
    if h>hbig
        h=hbig;
    end
else
    h=h/hcut;
    if h<hsmall
        error('Too small of a time step')
    end
end
nsteps=nsteps+1;
end
%Plotting solution with alpha = 4
plot(t3,y3,'g','LineWidth',2.0)
legend('\alpha=1','\alpha=2.5','\alpha=4','Location','nw')

```

### adaptive.m (Figure 7)

```

%Assigning global variables
global count
count=0;
global alpha
global beta
beta=(5/16);
[yinit, tinit,tfin, h, error, hraise,
hcut,hbig,hsmall]=textread('adaptive.dat','%f %f %f %f %f %f %f %f');
%Creating arrays and assigning initial values for the alpha = 1 graph
y1=[];
t1=[];
j=1;
y1(j)=yinit;
t1(j)=tinit;
nsteps=0;
alpha=1;
%Using the Adaptive Euler method for alpha = 1
while (t1<tfin)
    tnew=t1(j)+h;
    if tnew>tfin
        tnew=tfin;
        h=tnew-t1(j);
    end
    k0=yprime2(t1(j),y1(j));
    ylow=y1(j) + h*k0;
    k1=yprime2(tnew,ylow);
    yhigh=y1(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y1(j)=yhigh;
    end
end

```

```

        t1(j)=tnew;
        if h>hbig
            h=hbig;
        end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
    nsteps=nsteps+1;
end
%Creatting the graph for all the different data plots
figure(1)
plot(t1,y1,'r','LineWidth',2.0)
hold on
xlabel('Time (t)')
ylabel('y')
title('Adaptive model with Consumption for \alpha= 1, 2.5, and 4')
%Creating arrays and assigning initial values for the alpha = 2.5 graph
t2=[];
y2=[];
y2(j)=yinit;
t2(j)=tinit;
alpha=2.5;
%Using the Adaptive Euler method for alpha = 2.5
while (t2<tfin)
    tnew=t2(j)+h;
    if tnew>tfin
        tnew=tfin;
        h=tnew-t2(j);
    end
    k0=yprime2(t2(j),y2(j));
    ylow=y2(j) + h*k0;
    k1=yprime2(tnew,ylow);
    yhigh=y2(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y2(j)=yhigh;
        t2(j)=tnew;
        if h>hbig
            h=hbig;
        end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
    nsteps=nsteps+1;
end

plot(t2,y2,'c','LineWidth',2.0)

```

```

hold on

%Creating arrays and assigning initial values for the alpha = 4 graph
t3=[];
y3=[];
t3(j)=tinit;
y3(j)=yinit;
alpha=4;
%Using the Adaptive Euler method for alpha = 4
while (t3<tfin)
    tnew=t3(j)+h;
    if tnew>tfin
        tnew=tfin;
        h=tnew-t3(j);
    end
    k0=yprime2(t3(j),y3(j));
    ylow=y3(j) + h*k0;
    k1=yprime2(tnew,ylow);
    yhigh=y3(j)+((h/2)*(k0+k1));
    if abs((ylow-yhigh)/abs(yhigh))<error
        h=h*hraise;
        j=j+1;
        y3(j)=yhigh;
        t3(j)=tnew;
        if h>hbig
            h=hbig;
        end
    else
        h=h/hcut;
        if h<hsmall
            error('Too small of a time step')
        end
    end
    nsteps=nsteps+1;
end

```