Math 466 HW5 Directions & Guide

### HW5: Clayton Johnson

#### 10.1. Coffee cooling

Imagine you have a cup of coffee cooling according to the natural law of cooling, so that the temperature satisfies

where is a positive constant and is the temperature of the surroundings.

1. Assuming , separate the variables to obtain

First, let’s start with our differential equation,

and use the separation of variables technique to find the solution:

Since we know the initial condition (note that the function is a function of time ), then,

Substituting in and solving for ,

1. If the cup of coffee cools from an initial temperature to in 10 minutes, obtain and hence find the time it takes for the cup of coffee to cool to .

First, we will find , then we will solve for the time at which the temperature is , . To solve for , we use the equation found in the previous section:

Since we know that , we can find the value of :

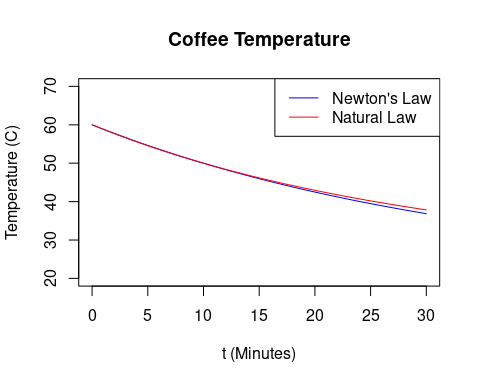
Now, with , we can solve for the time it takes for the temperature to drop to . Rearranging the previous equation, we solve for :

Using the values we’ve already defined (), in conjunction with the fact that , by definition, then we can calculate the time required:

1. Using Maple or MATLAB, compare the solution of this natural cooling model with the solution for Newton’s law of cooling by plotting them on the same system of axes, as in Figure 10.1. (Assume that with Newton’s law of cooling, the coffee cools from to in 10 minutes.)  
   Below is the code and output graph for this section. The code is using the same equations and values discussed previously in this section.

Ch101c <- function(T) {  
  
 #Chapter10.1 Model   
 #Perform Rk4 for temperature of coffee  
 #We find solutions for both Newton's law of cooling   
 #and for the natural law of cooling  
   
 #T is the time length in minutes for [0, T]  
 N <- 1000 #N is the number of time nodes  
 h <- T/N #This is the time step size in seconds  
   
 #System Parameters  
 t0 <- 0 #Initial time  
 u0 <- 60 #Initial temperature  
 us <- 20 #Temperature of surroundings  
 lambda1 <- 0.0288 #Newton's Law of Cooling (Ex10.2)  
 lambda2 <- 0.01186 #Natural Law of Cooling (Edit this value)  
 p <- 5/4 #Need to edit this for natural law of cooling; see below  
   
 #Slope functions for ODEs  
 f1 <- function(U) {-lambda1\*(U - us)} #Newton's law  
 f2 <- function(U) {-lambda2\*(U - us)^p} #Natural law  
   
 #Initialize vectors for time t and temperature U.   
 t <- rep(0, N) #Vector of N zeros (0 repeated N times)  
 U1 <- rep(0, N) #Vector of N zeros (0 repeated N times)  
 U2 <- rep(0, N) #Vector of N zeros (0 repeated N times)  
 t[1] <- t0 #Initial time t0  
 U1[1] <- u0 #Initial temperature  
 U2[1] <- u0 #Initial temperature  
  
   
 #Runge-Kutta Loop (Generate temperature vectors U1 & U2)  
 for(i in 1:N) {  
 a1 <- h\*f1(U1[i]); #f1 = slope of U1  
 a2 <- h\*f2(U2[i]); #f2 = slope of U2  
 b1 <- h\*f1(U1[i]+0.5\*a1); #Half-step predictor  
 b2 <- h\*f2(U2[i]+0.5\*a2); #Half-step predictor  
 c1 <- h\*f1(U1[i]+0.5\*b1); #Half-step predictor   
 c2 <- h\*f2(U2[i]+0.5\*b2); #Half-step predictor  
 d1 <- h\*f1(U1[i]+c1); #Full-step predictor  
 d2 <- h\*f2(U2[i]+c2); #Full-step predictor   
 U1[i+1] <- U1[i]+(1/6)\*(a1+2\*b1+2\*c1+d1); #Next U1 value  
 U2[i+1] <- U2[i]+(1/6)\*(a2+2\*b2+2\*c2+d2); #Next U2 value  
 t[i+1] <- t[i] + h  
 }  
   
 #Plot results  
 plot(t,U1,   
 main = "Coffee Temperature",  
 xlab = "t (Minutes)", #x-axis label  
 ylab = "Temperature (C)", #y-axis label  
 type = "l",col="blue", #Line type and color  
 xlim = c(0,T), #x-axis range vector: c(0,T) = [0,T]   
 ylim = c(20,70) #y-axis range vector: c = "combine"  
 )  
   
 lines(t,U2, type="l",col="red") #Place second graph in plot  
   
 legend("topright",  
 legend = c("Newton's Law", "Natural Law"), #Vector of legend items  
 col = c("blue","red"), #vector of colors for legend items  
 lty = c(1,1) #Vector of line types for legend items  
 )  
}

Ch101c(30)



#### 10.2 Beer warming.

A cold beer is at a temperature of . After 10 minutes, the beer has warmed to a temperature of . If the room temperature is , how long will it take the beer to warm to , assuming that Newton’s law of cooling applies?

In order to describe how the temperature of the beer changes over time (warming), assuming that Newton’s law of cooling applies, we can use the following equation from section 10.1 of the book:

Before solving for the time at which the beer’s temperature is , , we must solve for this term. First, let’s solve for :

Since we know that the beer is after 10 minutes (), we can use these values to find the numerical value for .

$$
\lambda = \frac{1}{10\text{ min}} \ln \left[ \frac{10\text{ }^{\circ}\text{C} - 30\text{ }^{\circ}\text{C}}{15\text{ }^{\circ}\text{C} - 30\text{ }^{\circ}\text{C}} \right], \\
\lambda = 0.02877\text{ min}^{-1}.
$$

Now, with the constant, we can solve for time, , at which the temperature is . Rearranging the previous equation to solve for time,

and substituting the values that we already know the values of (), we can find the time at which the beer’s temperature is :

$$
t\_{20} = \frac{ \ln \frac{10\text{ }^{\circ}\text{C} - 30\text{ }^{\circ}\text{C}}{20\text{ }^{\circ}\text{C} - 30\text{ }^{\circ}\text{C}}}{0.028877 \text{ min}^{-1}}, \\
t\_{20} = 24.09 \text{ min}.
$$

Below is the code and graph for the Beer Warming problem.

Ch102 <- function(T) {  
  
 #Chapter10.1 Model   
 #Perform Rk4 for temperature of coffee, assuming Newton's law of cooling  
   
 #T is the time length in minutes for [0, T]  
 N <- 1000 #N is the number of time nodes  
 h <- T/N #This is the time step size in seconds  
   
 #System Parameters  
 t0 <- 0 #Initial time  
 u0 <- 10 #Initial temperature  
 us <- 20 #Temperature of surroundings  
 lambda <- 0.02877 #Value of lambda  
   
 #Slope function from ODE  
 f <- function(U) {-lambda\*(U - us)}   
   
 #Initialize vectors for time t and temperature U.   
 t <- rep(0, N) #Vector of N zeros (0 repeated N times)  
 U <- rep(0, N) #Vector of N zeros (0 repeated N times)  
 t[1] <- t0 #Initial time t0  
 U[1] <- u0 #Initial temperature  
  
   
 #Runge-Kutta Loop (Generate temperature vector U)  
 for(i in 1:N) {  
 a1 <- h\*f(U[i]); #f = slope of U  
 b1 <- h\*f(U[i]+0.5\*a1); #Half-step predictor   
 c1 <- h\*f(U[i]+0.5\*b1); #Half-step predictor   
 d1 <- h\*f(U[i]+c1); #Full-step predictor   
 U[i+1] <- U[i]+(1/6)\*(a1+2\*b1+2\*c1+d1); #Next U value  
 t[i+1] <- t[i] + h  
 }  
   
 #Plot results  
 plot(t,U,   
 main = "Beer Temperature",  
 xlab = "t (Minutes)", #x-axis label  
 ylab = "Temperature (C)", #y-axis label  
 type = "l",col="blue", #Line type and color  
 xlim = c(0,T), #x-axis vector: c(0,T) = [0,T]   
 ylim = c(0,70) #y-axis vector: c = "combine" in R  
 )  
   
 legend("topright",  
 legend = c("Temperature"), #Vector of legend items  
 col = c("blue"), #vector of colors for legend items  
 lty = c(1) #Vector of line types for legend items  
 )  
}

Ch102(25)

