ModSim Exercise 7

Yoonyoung Cho

October 27 2015

Overview

The following MATLAB code oversees the entire assignment:

```
helper.m

%Part I
disp(simulate_4(1*60));
disp(simulate_4(5*60));
disp(simulate_4(10*60));

%Part II
for i = 0:30*60
    ans_vec(i+1) = simulate_4(i);
end
figure;
plot((0:30*60) / 60, ans_vec);
xlabel('Cream Addition Time(min)');
ylabel('Drinkable Time(sec)');
title('Drinkable Time per Cream Addition');

[val,time] = min(ans_vec);
disp(time);
%Part III
disp(fminsearch(@simulate_4,5.0*60));
```

Simulate_4 is a function that takes cream-addition time as an argument and returns the time at which the drinkable temperature is reached for the coffee.

In Part I, three values are investigated : the drinkable time as cream is added 1 minutes, 5 minutes, and 10 minutes after the simulation, respectively.

In Part II, the punchline plot is generated. This is done by measuring the drinkable time output from the cream-addition time (input) – ranging from 0 minutes(immediately) to 30 minutes – and then juxtaposing the result with the input time.

In Part III, the same simulation function undergoes the MATLAB function fminsearch in order to identify the exact optimal time at which to add the cream, as well as verifying the plot obtained from Part II.

1 Restructuring

Without further ado, simulate_4.m will be presented:

```
function drinkTime = simulate_4(addCreamTime)

%initial values
r = 8/2/100; %radius, m
```

```
h = 10/100; %height, m
A = pi*r^2+2*pi*r*h; %surface area, m^2
d = 0.7/100;%thickness, m
c = 4186; %specific heat, J/(kg*K)
rho = 1000; %density, kg/m^3
V = pi*r^2*h; %volume, m^3
m = rho*V; %mass, kg
k = 1.5; %heat conductivity, W/(m*K) => (in seconds)
T = 370;
T_e = 290;
E_0 = EbyT(T,m,c);

if(addCreamTime>0)
     zeroCross = @(T,E) zCross(T,E,m,c);%Event, reaching drinkable temperature.
 %passes additional parameters m,c (mass & specific heat) to compute the
 %temperature from energy.
    opt = odeset('Events',zeroCross);
    [t,E,alpha,beta,gamma] = ode45(@conduction,[0 addCreamTime],E_0,opt);

   if(length(alpha) ~= 0) % = there is a solution
        drinkTime = alpha;
        return;
   end

    [h,A,V,T,m,c,E_0] = addCream(h,r,V,m,c,E(end)); %update constants' values
    if(TbyE(E_0,m,c) < 320)% = adding cream resulted in drinkable mixture
        drinkTime = addCreamTime;
        return;
    end
else
     [h,A,V,T,m,c,E_0] = addCream(h,r,V,m,c,E_0); %update constants' values
    if(TbyE(E_0,m,c) < 320)% = adding cream resulted in drinkable mixture
        drinkTime = addCreamTime;
        return;
    end
end

%COFFEE Temperature simulation after adding Cream

 zeroCross = @(T,E) zCross(T,E,m,c); %Event, reaching drinkable temperature.
 %passes additional parameters m,c (mass & specific heat) to compute the
 %temperature from energy.
    opt = odeset('Events',zeroCross);
    [t,E,alpha,beta,gamma] = ode45(@conduction,[addCreamTime 30*60],E_0,opt);
    drinkTime = alpha; %assumption : the drinkable Time will be reached before
        30 min passes.
     return;

%conduction flow
    function Q = conduction(~,E)
        Q = -k*A*(TbyE(E,m,c)-T_e)/d;
    end

```

```
%simulating addition of cream
    function [h_new,A_new,V_new,T_new,m_new,c_new,E_new] = addCream(h,r,V,m,c,
        E)
        T = TbyE(E,m,c);

        %CREAM STUFF
        T_c = 275; %K
        V_c = 80/1e6;% m^3
        c_c = 4186; %specific heat, J/(kg*k)
        rho_c = 1000; %density, kg/m^3
        m_c = rho_c*V_c; %kg
        E_c = EbyT(T_c,m_c,c_c); %J

        %END OF CREAM STUFF
        %update constants
        h_new = h + V_c/(pi*r^2);
        A_new = pi*r^2+2*pi*r*h;
        V_new = V + V_c;
        T_new = (E+E_c)/(m*c+m_c*c_c);
        m_new = m + m_c;
        c_new = (c*m+m_c*c_c)/(m_new);
        E_new = E_c+E;
    end
end


function res = TbyE(E,m,c)
res = E/(m*c);
end

function res = EbyT(T,m,c)
res = T*m*c;
end

function [v,isT,dir] = zCross(t,E,m,c)
v = TbyE(E,m,c)-320;
isT = 1;
dir = 0;
end
```

The script is written such that no global variables, other than immutable constants, were used; the functions are encapsulated, and is coordinated in terms of logical association. For instance, the flow function is a part of the simulation – therefore, it was nested.

The simulation itself is composed of three phases : before, during, and after adding the cream. At any time, if the coffee temperature crosses the 320K (drinkable temperature) border, the function terminates and returns the time at which the event occurred.

The output values, for the investigation of discrete values, are as follows:

Table 1: Drinkable Time Investigation Table

$CreamAddition(min)$	$Drinkable(sec)$
1	291.0789
5	300
10	319.3362

2 Punchline Graph

It can be easily seen that the function above can be applied to any point in time – in whole units of seconds – at which to add the cream. Applying this to the interval from 0 to 30 minutes, the punchline graph below was generated:

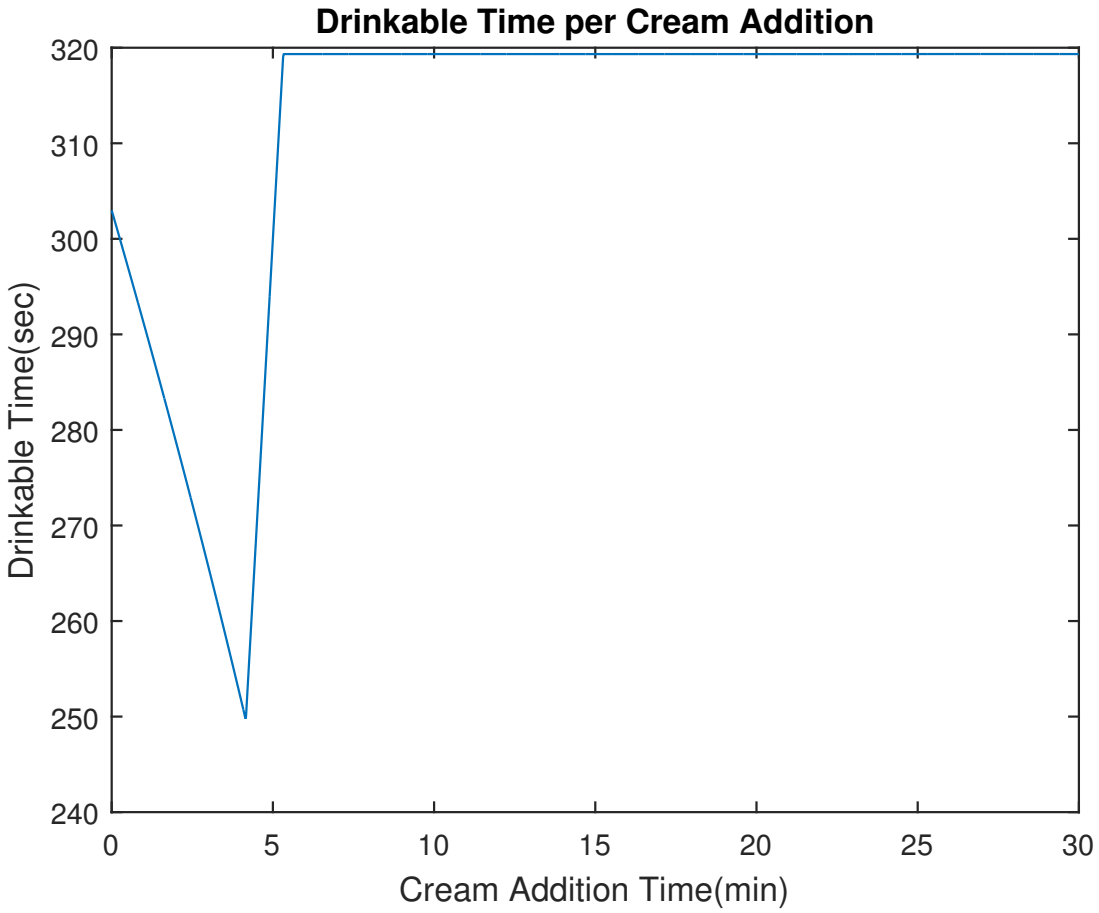

Figure 1: Part II Result: the plot of the time at which the coffee reached drinkable temperature (set to be 320 K) with respect to the time at which cream was inserted in the coffee.

The implications of this plots are consistent with intuition. It would be disadvantageous to insert the cream prematurely, as the heat flow is dependent on the difference between the temperature of the environment and the temperature of the system. Adding the cream would lower the temperature of the system, rendering it less efficient in terms of emitting energy to the environment.

However, it would be equally disadvantageous to stall for too long in adding the cream, as the coffee may

simply cool down below the drinkable temperature without the aid of the cream – which would, of course, take longer than optimal. It is thus evident that the ideal scenario would be to insert the cream – such that at the precise moment, the coffee would cool down to drinkable temperature as a result thereof.

The natural question, of course, is when that point is. By utilizing the MATLAB function min, which yields the index of the array at which the minimum value occurs (as its optional secondary return value), it was found that optimal value is 250 seconds (in resolutions of unit seconds, as the simulation only runs in seconds intervals).

3 Optimal Cream-Adding Time

The result above may be sufficient, but it is dissatisfying – the resolutions are limited in unit seconds – of course, this can be amended, but the alterations in the order of precision in the simulation would correspondingly incur a significant cost in time. Fortunately, I can use fminsearch in this case, with the starting value at 5 minutes, which is approximately where the plot reaches its trough.[1] The resultant output was 249.6337 seconds, which is definingly close to the simulated value. This therefore legitimizes the simulation.

[1] for reminders, the function was run as: $disp(fminsearch(@simulate\_4, 5.0 * 60))$.