%Name Nathan LeRoy 09/26/2018

%CLEAR THE VARIABLE SPACE:

%This block of code (Lines 10-12) is used to clear MATLAB of all variables

%and to close all currently open windows. This block of code does not

%represent anything physically in the model, it merely sets up the program

%for running this script. This block of code is included to prevent

%variable name mixups between scripts and to clean up the workspace.

clc %clears the command line

clear all %removes all variables from the workspace

close all %closes all windows in the program

%DEFINE THE PHYSICAL COMPOSITION OF OUR PUMPKIN PIE FILLING:

%This block of code (21-26) initializes mass fractions for each food component.

%All data on physical composition was found on the USDA food database. This

%data is necesssary as it allows us to calculate the relevant thermodynamic

%properties of our food item. Without them, no analysis on the temperature

%profile of our food over time could be done

X\_w = 0.48; %mass fraction of water in filling

X\_pro = 0; %mass frcation of protein in filling

X\_fat = 0; %mass fraction of fat in filling

X\_carb = 0.511; %mass fraction of carbohydrates in filling

X\_fib = 0.011; %mass fraction of fiber in filling

X\_ash = 0.0004; %mass fraction of ash in filling

%DEFINE INITIAL CONSTNATS AND CONDITIONS:

%This block of code (38-41) serves to set and define the initial condtions of

%our system. This is essential for solving our system numerically. We must

%know the initial temperature profile of our can and the tempereautre of

%the steam in order to begin transient anaylsis. Hewre we will also define

%the time and size steps in both dimensions for numerical analysis. This

%code block will also serve to initialize our matrix that will store and

%iterate through our data.

T\_pie\_o = 366.48; %Initial temperature of our filling, K

T\_steam = 121 + 273; %steam temperature in retort, K (121 deg C)

num\_slices = 10; %nuber of slices in finite difference method

num\_time\_steps = 1000; %number of time steps we wish to use

R = 2.125\*0.0254; %can radius in meters. Convert from 2.125 inches to meters.

Z = 6/100; %can height in meters. Convert from 6 inches to meters

t\_final = 10\*60\*60; %two hour final time, seconds

del\_t = t\_final/num\_time\_steps; %small time step for numerical analysis, seconds

del\_r = R/num\_slices; %small step in space in r-direction, m

%INITIALIZE OUR DATA MATRIX:

%This block of code (59-75) serves to initialize our matrix with the initial

%conditions of our system. Currently, the matrix is a zero matrix - we want

%to populate the entire thing with the initial temperature of our food

%product. Then, we want to initialize the outermost layer with our steam

%temeprature. This will serve as the starting point for the finite

%difference method.

%To optimize our code, it is necessary to predfine the size of our

%matrices. Otherwise, a large portion of our computational power is devoted

%to extending the memory of our matrices and can lead to sub-optimal

%performance when solving the finite difference method. We can do this by

%simply defining a zeros matrix that will be populated later.

T = zeros(num\_time\_steps,num\_slices+1);

for i = 1:1:num\_time\_steps %iteraete through each time step

for j = 1:1:length(T(1,1:num\_slices)) %iteraete through each slice (except for last one)

T(i,j) = T\_pie\_o; %set temperature to the initial temp.

end

end

%Initialize the outermost column to be the steam temperature

for i = 1:1:num\_time\_steps %iterate through each time point

T(i,num\_slices+1) = T\_steam; %set outermost slice/layer to the steam temperature.

end

%FINITE DIFFERENCE METHOD AND CALCULATION:

%This block of code (88-120) is the meat and potatoes of our algorithm. This nested

%loop will iterate through each time step and each slice, calculating the

%chage in temperature in both space adn time dimensions. Theoutermost loop

%iterates through each time point. At each time, we get to the next loop

%with iterates through each slice. The finite difference equation from

%geankopolis is used to get the temeprature gradient. Notice that the inner

%loop decrements. This is becasue the largest n is our outer-most slice,

%while an n of 1 corresponds to the center.

for i = 1:1:num\_time\_steps-1 %incrementing for loop

for n = num\_slices:-1:1 %decrementing for loop, Start at outside slice and move inward.

% Print statements for debugging

% fprintf('%d\n',n); %print statement for debug

% fprintf('Density of Pie Filling: %0.2f kg/m^3\n',rho\_pie);

% fprintf('Thermal Conductivity of Pie Filling: %0.2f W/m-K\n',k\_pie);

% fprintf('Heat Capacity of Pie Filling: %0.2f J/kg-K\n',cp\_pie);

temp = T(i,n)-273; %convert to celsius for thermal properties

therm\_constants = choi\_okos(X\_w,X\_pro,X\_fat,X\_carb,X\_fib,X\_ash,temp);

k\_pie = therm\_constants(1);

rho\_pie = therm\_constants(2);

cp\_pie = therm\_constants(3);

alpha = k\_pie/(rho\_pie\*cp\_pie); %thermal diffusivity of the pie filling

M = (del\_r^2)/(alpha\*del\_t); %define M constant for finite difference method

if(M<4) % stability check

error('M is not grater than 4, Unstable Solution ahead!'); %error stops script

end

if(n > 2) %Use generic finite difference equation for all points but the center

T(i+1,n) = (1/M)\*(((2\*n + 1)/(2\*n))\*T(i,n+1) + (M-2)\*T(i,n) + (2\*n - 1)/(2\*n)\*T(i,n-1));

else %Use the special case center-point equation.

T(i+1,n) = (4/M)\*T(i,n+1) + (M-4)/M\*T(i,n);

end

end

end

%CONVERT TO CELSIUS:

%Frequently, it is more intuitive and useful to view temperature gradients

%in units of relative temperature - like cesius or fahrenheit. This simple

%for loop and block of code (129-133) iterates through each row and column

%converting the temperature in degress Kelvin to the temperature in degrees

%celsius by subtracting 273 from the current temperature.

for i = 1:1:length(T(:,1));

for j = 1:1:length(T(1,:));

T(i,j) = T(i,j) - 273;

end

end

%CALCULATE LOG REDUCTION IN CAN:

%This code (146-169) allows us to calcualte the log reduction in the center of

%the can over time. It should be noted that we ASSUME a very long process,

%and then find the time-point where a 13.5 log reduction is achieved at the

%center. This time-point will be used as the heating process time and the

%temperature profile will be recalcualted. Without this recalculation, the

%cooling profile would be off and not accurate, as the material wuld be

%hotter than it is supposed to be. We choose to calculate the log-reduction

%here using the F0 method given our D250 value, and z-value. The units for

%this calculation are in seconds and Fahrenheit.

D250 = 0.2\*60; %seconds, assume largest in range 0.1-0.2 for maximum safety

z\_value = 12; %F

node = 1; % take at center

F\_0\_250 = 0; % intiialize at 0

F0\_vect = zeros(num\_time\_steps); %initialize with zeros

log\_red = zeros(num\_time\_steps,1); %initialize with zeros

for i = 1:1:num\_time\_steps-1 %iterate through all without exceeding matrix dimensions

temp = T(i,node)\*(9/5) + 32; %temp in Fahrenheit

F0\_vect(i+1) = F0\_vect(i) + del\_t\*10^((temp-250)/z\_value); %calciulate the F0

log\_red(i+1) = F0\_vect(i)/D250; %calcualte log reduction

end

for i = 1:1:num\_time\_steps %iterate through each

if log\_red(i) > 13.5 %is the log-reduction greater than 13.5?

time\_heat = (i-1)\*del\_t/60/60; %grab time

fprintf('A heating time of %0.2f hours produces a %0.2f log reduction\n',time\_heat,log\_red(i)); %print results

break %exit loop

end

end

% REDO CALCULATIONS WITH NEW HEATING TIME %

%DEFINE INITIAL CONSTNATS AND CONDITIONS AGAIN:

% We will be re-conducting the finite difference calculations (215-285) with our

% new-found heating time. If this was not done, we would have an over

% heated product which would affect our cooling and nutrient degredation

% caluclations.

R = 2.125\*0.0254; %can radius in meters. Convert from 2.125 inches to meters.

Z = 6/100; %can height in meters. Convert from 6 inches to meters

t\_final = time\_heat\*60\*60; %two hour final time, seconds

del\_t = t\_final/num\_time\_steps; %small time step for numerical analysis, seconds

del\_r = R/num\_slices; %small step in space in r-direction, m

%INITIALIZE OUR DATA MATRIX:

%This block of code (231-242) serves to initialize our matrix with the initial

%conditions of our system. Currently, the matrix is a zero matrix - we want

%to populate the entire thing with the initial temperature of our food

%product. Then, we want to initialize the outermost layer with our steam

%temeprature. This will serve as the starting point for the finite

%difference method.

T = zeros(num\_time\_steps,num\_slices+1);

for i = 1:1:num\_time\_steps %iteraete through each time step

for j = 1:1:length(T(1,1:num\_slices)) %iteraete through each slice (except for last one)

T(i,j) = T\_pie\_o; %set temperature to the initial temp.

end

end

%Initialize the outermost column to be the steam temperature

for i = 1:1:num\_time\_steps %iterate through each time point

T(i,num\_slices+1) = T\_steam; %set outermost slice/layer to the steam temperature.

end

%FINITE DIFFERENCE METHOD AND CALCULATION:

%This block of code (255-285) is the meat and potatoes of our algorithm. This nested

%loop will iterate through each time step and each slice, calculating the

%chage in temperature in both space adn time dimensions. Theoutermost loop

%iterates through each time point. At each time, we get to the next loop

%with iterates through each slice. The finite difference equation from

%geankopolis is used to get the temeprature gradient. Notice that the inner

%loop decrements. This is becasue the largest n is our outer-most slice,

%while an n of 1 corresponds to the center.

for i = 1:1:num\_time\_steps-1 %incrementing for loop

for n = num\_slices:-1:1 %decrementing for loop, Start at outside slice and move inward.

%fprintf('%d\n',n); %print statement for debug

% fprintf('Density of Pie Filling: %0.2f kg/m^3\n',rho\_pie);

% fprintf('Thermal Conductivity of Pie Filling: %0.2f W/m-K\n',k\_pie);

% fprintf('Heat Capacity of Pie Filling: %0.2f J/kg-K\n',cp\_pie);

temp = T(i,n)-273; %covnert to celsius for choi-okos

therm\_constants = choi\_okos(X\_w,X\_pro,X\_fat,X\_carb,X\_fib,X\_ash,temp);

%Extract constants%

k\_pie = therm\_constants(1);

rho\_pie = therm\_constants(2);

cp\_pie = therm\_constants(3);

alpha = k\_pie/(rho\_pie\*cp\_pie); %thermal diffusivity of the pie filling

M = (del\_r^2)/(alpha\*del\_t); %define M constant for finite difference method

if(M<4)

error('M is not grater than 4, Unstable Solution ahead!'); %error ends script

end

if(n > 2) %Use generic finite difference equation for all points but the center

T(i+1,n) = (1/M)\*(((2\*n + 1)/(2\*n))\*T(i,n+1) + (M-2)\*T(i,n) + (2\*n - 1)/(2\*n)\*T(i,n-1));

else %Use the special case center-point equation.

T(i+1,n) = (4/M)\*T(i,n+1) + (M-4)/M\*T(i,n);

end

end

end

%CONVERT TO CELSIUS:

%Frequently, it is more intuitive and useful to view temperature gradients

%in units of relative temperature - like cesius or fahrenheit. This simple

%for loop and block of code (294-298) iterates through each row and column

%converting the temperature in degress Kelvin to the temperature in degrees

%celsius by subtracting 273 from the current temperature.

for i = 1:1:length(T(:,1));

for j = 1:1:length(T(1,:));

T(i,j) = T(i,j) - 273;

end

end

%DEFINE AND INITIALIZE A MATRIX TO STORE THE COOLING TEMPERATURE:

%After we complete and model our heating process, we need to complete and

%model the cooling process. Here we imagine our cans are being submerged in

%a bath of water with a constant temperature of 55 degrees C. Our cans

%leave the retort with a temperature profile identical to the one they had

%at the end of the heating process. Thus, our cooling temeprature matrix is

%identical to our heating matrix, with the exception of the outermost layer

%temperature being 55 degrees C instead of 121 degrees C.

%Lines (311-320)

T\_cool = zeros(num\_time\_steps,num\_slices+1);

T\_cool(:,num\_slices+1) = 12.7;

T\_cool(1,:) = T(num\_time\_steps,:);

%convert T\_cool to kelvin for analysis

for i = 1:1:length(T\_cool(:,1));

for j = 1:1:length(T\_cool(1,:));

T\_cool(i,j) = T\_cool(i,j) + 273;

end

end

%FINITE DIFFERENCE METHOD AND CALCULATION:

%We must now run the fnite difference model again with our new outer

%temperature. For simplicity, the exact same assumptions are being made and

%this, we can use the same equations. Our cans are now just being emersed

%in a cool nath of water. We run this cooling process for the same time as

%our heating process. We will then extract the time the average temperature

%in the can is 100F

%Lines(331-377).

for i = 1:1:num\_time\_steps-1 %incrementing for loop

for n = num\_slices:-1:1 %decrementing for loop, Start at outside slice and move inward.

%fprintf('%d\n',n); %print statement for debug

% fprintf('Density of Pie Filling: %0.2f kg/m^3\n',rho\_pie);

% fprintf('Thermal Conductivity of Pie Filling: %0.2f W/m-K\n',k\_pie);

% fprintf('Heat Capacity of Pie Filling: %0.2f J/kg-K\n',cp\_pie);

temp = T\_cool(i,n)-273;

therm\_constants = choi\_okos(X\_w,X\_pro,X\_fat,X\_carb,X\_fib,X\_ash,temp);

k\_pie = therm\_constants(1);

rho\_pie = therm\_constants(2);

cp\_pie = therm\_constants(3);

alpha = k\_pie/(rho\_pie\*cp\_pie); %thermal diffusivity of the pie filling

M = (del\_r^2)/(alpha\*del\_t); %define M constant for finite difference method

if(M<4)

error('M is not grater than 4, Unstable Solution ahead!');

end

if(n > 2) %Use generic finite difference equation for all points but the center

T\_cool(i+1,n) = (1/M)\*(((2\*n + 1)/(2\*n))\*T\_cool(i,n+1) + (M-2)\*T\_cool(i,n) + (2\*n - 1)/(2\*n)\*T\_cool(i,n-1));

else %Use the special case center-point equation.

T\_cool(i+1,n) = (4/M)\*T\_cool(i,n+1) + (M-4)/M\*T\_cool(i,n);

end

% if(n == num\_slices) %Use generic finite difference equation for all points but the center

%

% elseif(n>2 && < num\_slices %Use the special case center-point equation.

% T(i+1,n) = (1/M)\*(((2\*n + 1)/(2\*n))\*T(i,n+1) + (M-2)\*T(i,n) + (2\*n - 1)/(2\*n)\*T(i,n-1));

% else

% T(i+1,n) = (4/M)\*T(i,n+1) + (M-4)/M\*T(i,n);

% end

end

end

%convert T\_cool to celsius for analysis

for i = 1:1:length(T\_cool(:,1));

for j = 1:1:length(T\_cool(1,:));

T\_cool(i,j) = T\_cool(i,j) - 273;

end

end

%Concatenate our heating matrix with our cooling matrix. This allows us to

%manipulate and calcualte data for the entire process instead of using two

%separate matrices.

T\_process = [T;T\_cool]; %concatenate on top of each other.

avg\_temp = zeros(2\*num\_time\_steps,1);

%CALCUALTE AVERAGE TEMPERATURE:

%This block of code(391-408) will calcualte the average temperature in our can for

%each time-point. This is done by weighting the temperature in each slice

%by its relative volume in the can as awhole, and then summing these

%temperatures up. It is the discrete version of the average value theorem.

%it also looks for and fidns the time the average temperature is the

%required exit temperature in our can.

for i=1:1:2\*num\_time\_steps %run each time step

for j = num\_slices:-1:1 %decrement from edge to cetner

avg\_temp(i) = avg\_temp(i) + ((1/R)^2)\*T\_process(i,j)\*((j\*del\_r)^2 - ((j-1)\*del\_r)^2); %calcuatle avg temp

end

end

j=0; %create counter

for i = num\_time\_steps:1:2\*num\_time\_steps

if avg\_temp(i) < 37.77 %search for an average temp of 100 F

time\_cool = (j)\*del\_t/60/60;

fprintf('A cooling time of %0.2f hours leaves contents of can at an average temperature of %0.2f F\n',time\_cool,avg\_temp(i)\*9/5 + 32); %print temp in fahrenheit

break %exit loop

end

j = j + 1; %increment counter

end

%CALCULATE LOG REDUCTION IN CAN:

%This code (421-433) allows us to calcualte the log reduction in the center of

%the can over time. It should be noted that we ASSUME a very long process,

%and then find the time-point where a 13.5 log reduction is achieved at the

%center. This time-point will be used as the heating process time and the

%temperature profile will be recalcualted. Without this recalculation, the

%cooling profile would be off and not accurate, as the material wuld be

%hotter than it is supposed to be. We choose to calculate the log-reduction

%here using the F0 method given our D250 value, and z-value. The units for

%this calculation are in seconds and Fahrenheit.

D250 = 0.2\*60; %seconds, assume largest in range 0.1-0.2 for maximum safety

z\_value = 12; %F

node = 1; % take at center

F\_0\_250 = 0;

F0\_vect = zeros(2\*num\_time\_steps);

log\_red = zeros(2\*num\_time\_steps,1);

for i = 1:1:2\*num\_time\_steps-1

temp = T\_process(i,node)\*(9/5) + 32; %temp in Fahrenheit

F0\_vect(i+1) = F0\_vect(i) + del\_t\*10^((temp-250)/z\_value);

log\_red(i+1) = F0\_vect(i)/D250;

end

%CALCULATE VITAMIN B1 LOG REDUCTION IN CAN:

%This code (441-454) allows us to calcualte how much of vitamin B1 has been

%destroyed over time. The log-reduction specifically. This is done for each

%slice at each time point. The process is identical to the log-reduction of

%microorganisms in our can over time.

D250\_v = 246.9\*60; %seconds, assume largest in range 0.1-0.2 for maximum safety

z\_value\_v = 49; %F

node = num\_slices; % take at the edge

F0v\_vect = zeros(2\*num\_time\_steps,num\_slices); %initialize vector

log\_red\_v = zeros(2\*num\_time\_steps,num\_slices); %initialize vector

for i = 1:1:2\*num\_time\_steps-1 %iterate through entire matrix time points

for j = num\_slices:-1:1 %iterate from outermost slice to center

temp = T\_process(i,j)\*9/5 + 32; %temperature in fahrenheit

F0v\_vect(i+1,j) = F0v\_vect(i,j) + del\_t\*10^((temp - 250)/z\_value\_v);

log\_red\_v(i+1,j) = F0v\_vect(i,j)/D250\_v; %calculate log reduction

end

end

%CALCUALTE AVERAGE LOG REDUCTION of B1:

%This block of code(391-408) will calcualte the average log-reduction in our can for

%each time-point. This is done by weighting the log-red in each slice

%by its relative volume in the can as awhole, and then summing these

%temperatures up. It is the discrete version of the average value theorem..

avg\_log\_red\_v = zeros(2\*num\_time\_steps,1); %intiialize vector

for i=1:1:2\*num\_time\_steps % iterate through each time-point

for j = num\_slices:-1:1 %iterate from outermost slice to center

avg\_log\_red\_v(i) = avg\_log\_red\_v(i) + ((1/R)^2)\*log\_red\_v(i,j)\*((j\*del\_r)^2 - ((j-1)\*del\_r)^2); %calcualte average

end

end

%convert our log reduction to a percentage reduction

avg\_log\_red\_v\_perc = ones(length(avg\_log\_red\_v),1);

for i = 1:1:length(avg\_log\_red\_v)

avg\_log\_red\_v\_perc(i) = (1/(10^avg\_log\_red\_v(i)))\*100;

end

%CALCULATE THE VITAMIN C REDUCTION IN THE CAN

%identical to VITAMIN B1 reduction, but use different comments.

D250\_c = 1.12\*24\*60\*60; %seconds, assume largest in range 0.1-0.2 for maximum safety

z\_value\_c = 52; %F

node = num\_slices; % take at the edge

F0c\_vect = zeros(2\*num\_time\_steps,num\_slices);

log\_red\_c = zeros(2\*num\_time\_steps,num\_slices);

for i = 1:1:2\*num\_time\_steps-1

for j = num\_slices:-1:1

temp = T\_process(i,j)\*9/5 + 32; %temperature in fahrenheit

F0c\_vect(i+1,j) = F0c\_vect(i,j) + del\_t\*10^((temp - 250)/z\_value\_c);

log\_red\_c(i+1,j) = F0c\_vect(i,j)/D250\_c;

end

end

%Initialize vector to store average value

avg\_log\_red\_c = zeros(2\*num\_time\_steps,1);

%calcualte the average log-reduction at each

%timepoint

for i=1:1:2\*num\_time\_steps %iterate through each time-point

for j = num\_slices:-1:1 %iterate from outermost slice to center

avg\_log\_red\_c(i) = avg\_log\_red\_c(i) + ((1/R)^2)\*log\_red\_c(i,j)\*((j\*del\_r)^2 - ((j-1)\*del\_r)^2); %caluclate the average log-reduction

end

end

%convert average log-reduction to a percent reduction

avg\_log\_red\_c\_perc = ones(length(avg\_log\_red\_v),1);

for i = 1:1:length(avg\_log\_red\_v)

avg\_log\_red\_c\_perc(i) = (1/(10^avg\_log\_red\_c(i)))\*100;

end

dq = zeros(num\_time\_steps,1); %initialize the vector to store energy chagne

%CALCULATE ENREGY REQUIREMENTS:

%Lines (521-531) Using the discrete energy change equation Q=m\*cp\*delT to calcualte the

%change in energy inside our cans. This is useful for calcualting the

%economics of oru process. The choi okos equation is required to calculate

%the thermal properties in our can.9

for i = 1:1:num\_time\_steps

temp = avg\_temp(i);

therm\_constants = choi\_okos(X\_w,X\_pro,X\_fat,X\_carb,X\_fib,X\_ash,temp);

k\_pie = therm\_constants(1);

rho\_pie = therm\_constants(2);

cp\_pie = therm\_constants(3);

dq(i) = rho\_pie\*pi\*R^2\*Z\*cp\_pie\*(temp-(T\_pie\_o-273));

end

%PLOT THE DATA:

%This section of code (541-593) serves to only present the data visually to the user.

%This is imperitive to help the user visualize what is occuring inside our

%can over time and amake education decisions on what the optimized

%paramters and steps need to be.

time\_fig = [0:del\_t/60/60:2\*num\_time\_steps\*del\_t/60/60 - del\_t/60/60];

dist = [0:del\_r\*1000:R\*1000];

%TEMPERATURE PROFILE AT VARIOUS TIMES

figure(1)

hold on

plot(dist,T(1,1:num\_slices+1),'-k');

plot(dist,T(floor((t\_final/4)/del\_t + 1),1:num\_slices+1),'b-');

plot(dist,T(floor((t\_final/2)/del\_t + 1),1:num\_slices+1),'g-');

plot(dist,T(floor((3\*t\_final/4)/del\_t + 1),1:num\_slices+1),'r-');

legend('t = 0','t = 25% processing time', 't = 50% processing time', 't = 75% processing time','location','northwest');

title(['Temperature profile in can ',num2str(t\_final/60/60),' hr'])

xlabel('Distance from Center (mm)');

ylabel('Temperature (deg C)');

legend('boxoff');

%LOG REDUCTION IN C.BOT OVER TIME

figure(2)

hold on

plot(time\_fig,log\_red);

title('C. Botulinum Destruction | Center of Can')

ylabel('log-reduction');

xlabel('Time (hrs)');

legend('C. Botulinum','location','northwest');

legend('boxoff');

%TEMPERATURE OVER TIME AT VARIOUS LOCATIONS IN OUR CAN

figure(3)

hold on

plot(time\_fig,T\_process(:,1),'-r');

plot(time\_fig,T\_process(:,num\_slices),'b-');

plot(time\_fig,T\_process(:,floor(num\_slices/2)),'g-');

plot(time\_fig,avg\_temp);

title('Full Process Temperature Profile');

xlabel('Time (Hr)');

ylabel('Temperature (deg C)');

legend('0 mm','r = 50mm','r = 25 mm','Average','location','northeast');

legend('boxoff');

%VITAMIN DESTRUCTION OVER TIME

figure(4)

hold on

plot(time\_fig,avg\_log\_red\_v\_perc,'b-');

plot(time\_fig,avg\_log\_red\_c\_perc,'r-');

ylabel('% Nutrients in Can');

ylim([0 100]);

xlabel('Time (hrs)');

title('% Nutrient in Can');

legend('Vitamin B1','Vitamin C','location','southwest');

legend('boxoff');

%ENERGY REQUIRMENTS/ABSORPTION IN CAN

figure(5)

hold on

plot(time\_fig(1:num\_time\_steps),dq./1000);

ylabel('Energy, kJoules');

xlabel('Time (hrs)');

title('Energy Absorption per Can');

%FUNCTION | CHOI - OKOS EQUATION:

%This function serves to calculate three thermodynamic properties of a food

%component given the mass fraction of specific items in the food, and the

%operating temperature. Specifically, it returns the value of the density

%of a food item, the thermal conductivity of the food item, and the heat

%capacity of the food item.

%INPUTS: X\_w - mass fraction of water

% X\_pro - mass fraction of protein

% X\_fat - mass fraction of fat

% X\_carb - mass fraction of carbohydrates

% X\_fib - mass fraction of fiber

% X\_ash - mass fraction of ash

%OUTPUTS:

% therms - a matrix storing the values: [k, rho, cp]

%

function therms = choi\_okos(X\_w, X\_pro, X\_fat, X\_carb, X\_fib, X\_ash,temp\_operation)

rho\_w = 9.9718e2 + 3.1439e-3\*temp\_operation - 3.7574e-3\*temp\_operation^2; %desnity of water, kg/m^3

rho\_pro = 1.3299e3 - 5.1840e-1\*temp\_operation; %desnity of protein, kg/m^3

rho\_fat = 9.2559e2 - 4.1757e-1\*temp\_operation; %density of fat, kg/m^3

rho\_carb = 1.5991e3 - 3.1046e-1\*temp\_operation; %density of carbohydrates, kg/m^3

rho\_fib = 1.3115e3 - 3.6589-1\*temp\_operation; %density of fiber, kg/m^3

rho\_ash = 2.4238e3 - 2.8063e-1\*temp\_operation; %density of ash, kg/m^3

k\_w = 5.7109e-1 + 1.7625e-3\*temp\_operation - 6.7036e-6\*temp\_operation^2; %therm. cond. water, W/m-K

k\_pro = 1.7881e-1 + 1.1958e-3\*temp\_operation - 2.7178e-6\*temp\_operation^2; %therm. cond. protein W/m-K

k\_fat = 1.8071e-1 + 2.7604e-3\*temp\_operation - 1.7749e-6\*temp\_operation^2; %th erm. cond. fat W/m-K

k\_carb = 2.0141e-1 + 1.3874e-3\*temp\_operation - 4.3312e-6\*temp\_operation^2; %therm. cond. carb W/m-K

k\_fib = 1.8331e-1 + 1.2497e-3\*temp\_operation - 3.1683e-6\*temp\_operation^2; %therm. cond. fiber W/m-K

k\_ash = 3.2962e-1 + 1.4011e-3\*temp\_operation - 2.9069e-6\*temp\_operation^2; %therm. cond. ash W/m-K

cp\_w = 4.1762e3 - 9.0864e-2\*temp\_operation + 5.4731e-3\*temp\_operation^2; %heat capacity of water, J/kg-K

cp\_pro = 2.0082e3 + 1.2089\*temp\_operation - 1.3129e-3\*temp\_operation^2; %heat capacity of protein, J/kg-K

cp\_fat = 1.9842e3 + 1.4733\*temp\_operation - 4.8008e-3\*temp\_operation^2; %heat capacity of fat, J/kg-K

cp\_carb = 1.5488e3 + 1.9625\*temp\_operation - 5.9399e-3\*temp\_operation^2; %heat capacity of carbs, J/kg-K

cp\_fib = 1.8459e3 + 1.8306\*temp\_operation - 4.6509e-3\*temp\_operation^2; %heat capacity of fiber., J/kg-K

cp\_ash = 1.0926e3 + 1.8896\*temp\_operation - 3.6817e-3\*temp\_operation^2; %heat capcity of ash, J/kg-K

%calculate the thermal properites of our food according to the choi - okos

%equation

k = X\_w\*k\_w + X\_pro\*k\_pro + X\_fat\*k\_fat + X\_carb\*k\_carb + X\_fib\*k\_fib + X\_ash\*k\_ash; %W/m-K

rho = X\_w\*rho\_w + X\_pro\*rho\_pro + X\_fat\*rho\_fat + X\_carb\*rho\_carb + X\_fib\*rho\_fib + X\_ash\*rho\_ash; %kg/m^3

cp = X\_w\*cp\_w + X\_pro\*cp\_pro + X\_fat\*cp\_fat + X\_carb\*cp\_carb + X\_fib\*cp\_fib + X\_ash\*cp\_ash; %J/kg-K

therms = [k,rho,cp];

end