Project

Wednesday, November 20, 2024 10:19 PM

EME 165 Numerical Conduction Project

Assigned: Sunday, Nov 10th 2024
Due: Saturday, Dec 7th; by 11:59 pm
Firm deadline- no extensions
Points allocated: 15

1. This is an individual project. Derivations and coding should be the sole work of the individual submitting the work. Discussions with TA, faculty and peers in the class are permitted. Use of textbook is also permitted. Use of generative AI to debug the code is allowed. The student should clearly note how this resource was used in the submission. However, no online tutors (for example, tutors on sites like chegg.com) or other sources for any other part such as formulation of the project may be used.
2. You are required to use Matlab or a programming language to solve the problem. Use of *Excel, or a commercially available solver is not permitted.*
3. Begin your project as soon as it is assigned. Do not wait until the last minute. Formulation and debugging of the code can take longer than you may expect.
4. Submit a single pdf of derivation/formulation and the code and solution outputs in gradescope. In a separate gradescope assignment "Project 1 code" submit your code.

Pledge (Project will not be graded without a signed student pledge):

I confirm that my work in this project is **solely my own effort** and that I have not received assistance from **any person OR organization within or outside the class.** I understand that while discussion with friends is permitted, but the formulation and coding are my own work. I furthermore understand that any impropriety will be reported to the UC Davis student misconduct office for disciplinary action.

Student Signature: [signature]

If you are using blank sheets of paper for the solution rather than this printout, acknowledge that you have read the pledge above by signing your name such as:
Pledge: I have read and accept the pledge.
Signature: [signature]

Problem statement: Post-harvest refrigeration

Post-harvest refrigeration is essential for extending the shelf life of fruits and vegetables. After a cantaloupe has ripened to a desired level, it needs to be properly refrigerated to keep it fresh during its long-distance transport from the field to the grocery store.

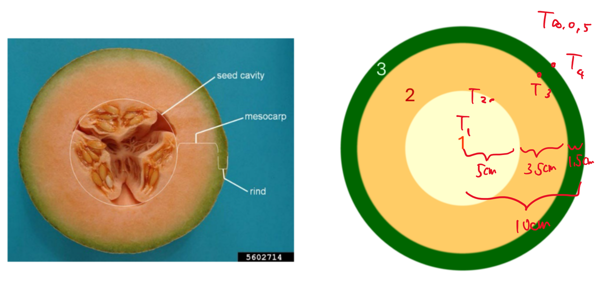
Determine the time and energy requirement for the post-harvest refrigeration of a cantaloupe using the information given in the problem.

Figure 1 shows the cross-section of a cantaloupe. It has three different regions: i) the rind: outer shell, ii) the fruit region, and iii) the seed-cavity. The dimensions and thermo-physical properties of the different sections of a cantaloupe are provided in Table 1.

The radius of the seed cavity is 5 cm, followed by a fruity-flesh region (mesocarp) of thickness 3.5 cm, followed by an outer shell/rind of thickness 1.5 cm. Assume that a cantaloupe can be modeled as a sphere of radius 10 cm.

Table 1: Dimensions and properties of different regions of a cantaloupe.

Regions	Radius (cm)	Thickness (cm)	Thermal conductivity (W/m-K)	Specific heat capacity (J/kg-K)	Density (kg/m³)
Outer shell/rind (3)	-	1.5	0.2	3150	1200
Fruity flesh /mesocarp(2)	-	3.5	0.6	4200	1000
Seed cavity (1)	5	-	0.03	1470	520



(a) (b)
Figure 1: Cross-section of a cantaloupe: (a)-actual fruit's cross-section (source: internet) (b) modeled cross-section

The cantaloupe is cooled inside an industrial refrigeration storage facility via forced air convection. The temperature of the cold air is 2 °C, and convective heat transfer coefficient at the cantaloupe shell-air interface is 50 W/m²-K, and heat is transported into the inner surface of the fruits via conduction.

The initial temperature of the harvested cantaloupe is 30°C, and the cooling process is complete when the temperature of the fruity-flesh region at the seed-cavity interface reaches 5 °C. when $T_2 = 5°C$

a. Determine whether the transient temperature variation of the cantaloupe can be estimated using a lumped capacitance model. Explain the reasons for your answer.

b. Using 1-D approximate solution to transient conduction, determine how long it will take for the temperature of the inner radius of the fruity flesh region to reach 5 °C.

Hint: Use mass weighted average properties to form a hypothetical solid cantaloupe with uniform properties.

c. Numerically simulate the cooling process and determine the time (final time) it will take for the temperature of the fruit-seed-cavity interface to reach 5 °C. Use a minimum of 5 nodes in the seed cavity, and a minimum of 5 nodes in the fruit flesh regions in the radial direction. Plot the temperature distribution (T vs r) within the cantaloupe at 5 times including the initial time and the final time.

d. Calculate the energy required (in Joules) to cool the fruit to this temperature distribution by refrigeration system

e. Compare your answer from parts b and c, and comment on any differences you have observed.

BONUS (1 point): Vary the convective heat transfer coefficient from 50 W/m²-K to 180 W/m²-K. Determine the time taken for the fruit-seed cavity interface to reach 5 °C. Plot the time on the y-axis vs heat transfer coefficient on the x-axis.

You have to design an industrial post-harvest refrigeration facility to cool 500 cantaloupes, as shown in *Figure 2*. You have decided to use the same forced convection air-based cooling technology to cool 500 cantaloupes in your industrial refrigeration facility.

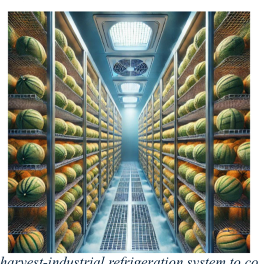
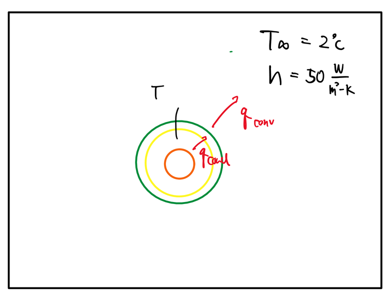


Figure 2: Post-harvest-industrial refrigeration system to cool cantaloupes.

f. Using your answer from parts c and d, try to infer the time and energy it will take to cool 500 cantaloupes. Identify and explain if you will have to include any additional factors when you determine the cooling load for your industrial refrigeration system.

Grading Scheme

Parts	Points
a. LC method question	0.5
b. 1-D transient conduction- analytical solution	2
c. Numerical conduction	
• Schematic of the control volumes and nodes, identify unique nodes	1.5
• Derivation of the energy balance equations for unique nodes	5
• Working code	3
• Plot and numerical answer	1
d. energy needed to cool	1
e. Compare analytical and numerical solutions	0.5
f. refrigeration system estimate	0.5
BONUS question	1
Total	**15 +1 bonus**



a) The outermost layer:

The Biot #: $Bi = \frac{h \cdot L_c}{k}$

$L_c = \frac{V}{A_s} = \frac{\frac{4\pi}{3}(0.1^3 - 0.085^3)}{4\pi \cdot 0.1^2} = 0.0128625$

$Bi = \frac{50 \cdot 0.0128625}{0.2} = 3.215625$

$3.21 \gg Bi = 0.1$

⇒ Lumped capacitance method would not work

b)

$m_{total} = \rho_1 V_1 + \rho_2 V_2 + \rho_3 V_2$

$= 520 \cdot \underbrace{\frac{4}{3}\pi \cdot 0.05^3}_{0.0005236} + 1000 \cdot \underbrace{\frac{4}{3}\pi(0.085^3 - 0.05^3)}_{0.00205} + 1200 \cdot \underbrace{\frac{4}{3}\pi(0.1^3 - 0.085^3)}_{0.001616}$

$= 0.2723 + 2.0488 + 1.9396$

$= 4.26073$ kg

$C_{p,total} = \frac{m_1}{m_{tot}} c_{p,1} + \frac{m_2}{m_{tot}} c_{p,2} + \frac{m_3}{m_{tot}} c_{p,3}$

$= 3547.5533\ J/kg \cdot K$

$m = \rho \cdot V \Rightarrow \rho = \frac{m_1}{V_1} + \frac{m_2}{V_2} + \frac{m_3}{V_3}$

$\rho_{total} = \rho_1 \cdot \frac{V_1}{V} + \rho_2 \cdot \frac{V_2}{V} + \rho_3 \cdot \frac{V_3}{V}$

$= 520 \cdot \frac{0.00052}{0.0041896} + 1000 \cdot \frac{0.00205}{0.0041896} + 1200 \cdot \frac{0.00166}{0.0041896}$

$= 1017.1549\ kg/m^3$

$k_{total} = k_1 \cdot \frac{m_1}{m_T} + k_2 \cdot \frac{m_2}{m_T} + k_3 \cdot \frac{m_3}{m_T}$

$= 0.03 \cdot \frac{0.2723}{4.26073} + 0.6 + \frac{2.0488}{4.26073} + 0.2 \cdot \frac{1.9396}{4.26073}$

$= 0.38147$

New Biot number: $Bi = \frac{h \cdot L_c}{k}$ under mass weighted average ⇒ as a whole solid

$= \frac{50 \cdot 0.1}{0.38147}$

$= 13.107 \Rightarrow Bi^{-1} = 0.07629$

$\theta_o^* = \frac{T_o - T_\infty}{T_i - T_\infty} = \frac{5 - 2}{30 - 2} = 0.10714$

From Table 5.1, $Bi = 13.107$ $\xrightarrow[\text{interpolation}]{\text{Sphere}}$ $\zeta_1 = 2.8363 + \frac{2.9857 - 2.8363}{10} \cdot 3.107$

$= 3.0297$

$C_1 = 1.9249 + \frac{1.9781 - 1.9249}{10} \cdot 3.107$

$= 1.9414$

$\theta^* = \frac{T - T_\infty}{T_i - T_\infty} = \theta_o^* \cdot \frac{1}{\zeta_1 \cdot r^*}(\sin(\zeta_1 \cdot r^*))$ $r^* = \frac{r}{r_o}$

$\theta_o^* = C_1 \exp(-\zeta_1^2 \cdot F_o)$

$0.10714 = 1.9414 \exp(-(3.0297)^2 \cdot F_o)$

$F_o = 0.315612 > 0.2 \Rightarrow$ 1D approximate solutions

$F_o = \frac{\alpha t}{L_c^2} = \frac{k}{\rho \cdot c_p} \cdot \frac{t}{L_c^2}$ $L_c = 0.1$

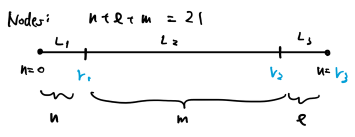
$0.315612 = \frac{0.38147}{1017.155 \cdot 3547.5533} \cdot \frac{t}{0.1^2}$

$t = 29811.921$ sec

↓

$t = 8.281$ hrs

c)

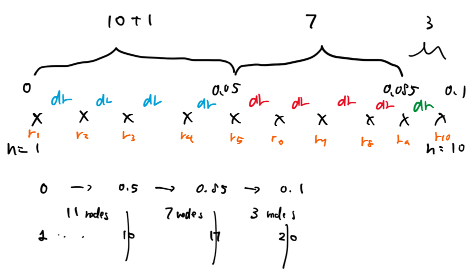


Nodes: $n + l + m = 21$

dr = 0.05 / (n − 1); dr = layer thickness / (m − 1); dr = layer thickness / (l − 1)

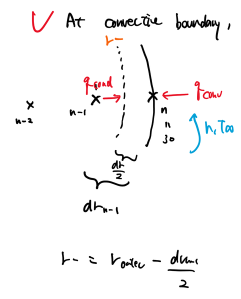
nodes per layer $= 21 \cdot \frac{L_1}{L_{total}} / \frac{L_2}{L_{total}} / \frac{L_3}{L_{total}}$

$r_1 = dr \cdot n$



0 → 0.5 → 0.85 → 0.1
11 nodes | 7 nodes | 3 nodes
1 … 10 | 17 | 20

✓ At convective boundary,



$r_- = r_{outer} - \frac{dr_{n-1}}{2}$

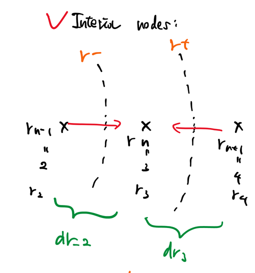
$q_{n-1 \to n} + q_{conv} = \rho \cdot c_p \cdot dV \cdot \frac{T_n^{p+1} - T_n^p}{dt}$

$q_{n-1 \to n} + q_{conv} = \frac{\rho c_p dV}{dt} \cdot (T_n^{p+1} - T_n^p)$

$\frac{3k \cdot (r_o - \frac{dr}{2})^2}{dr} \cdot (T_{n-1}^p - T_n^p)dt + 3h \cdot r_o^2 (T_\infty - T_n^p)dt = \rho c_p \cdot (r_o^3 - (r_o - \frac{dr}{2})^3) \cdot \frac{1}{dt}(T_n^{p+1} - T_n^p)$

$\frac{3k \cdot (r_o - \frac{dr}{2})^2 dt}{dr \cdot \rho \cdot c_p \cdot (r_o^3 - (r_o - \frac{dr}{2})^3)}(T_{n-1}^p - T_n^p) + \frac{3h r_o^2 dt}{\rho \cdot c_p (r_o^3 - (r_o - \frac{dr}{2})^3)} \cdot (T_\infty - T_n^p) = T_n^{p+1} - T_n^p$

✓ Interior nodes:



$r_+ = r_3 + \frac{dr_3}{2}$

$r_- = r_3 - \frac{dr_3}{2}$

$q_{n-1 \to n} + q_{n+1 \to n} = \rho \cdot c_p \cdot dV \frac{T_n^{p+1} - T_n^p}{dt}$

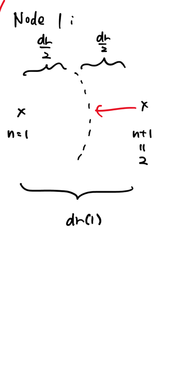
$q_{n-1 \to n} + q_{n+1 \to n} = \frac{\rho c_p dV}{dt} \cdot (T_n^{p+1} - T_n^p)$

$\frac{k \cdot (r_-)^2}{dr}(T_{n-1}^p - T_n^p) + \frac{k \cdot (r_+)^2}{dr}(T_{n+1}^p - T_n^p) = \left(\frac{\rho \cdot c_p \cdot (r_n^3 - r_-^3)}{3\,dt} + \frac{\rho \cdot c_p (r_+^3 - r_n^3)}{3\,dt}\right)(T_n^{p+1} - T_n^p)$

k_{n-1} k_{n+1} $\rho_n, c_{p,n}$

k_1 | 0.05 | k_2 | 0.085 | k_3 | 0.10

✓ Node 1:



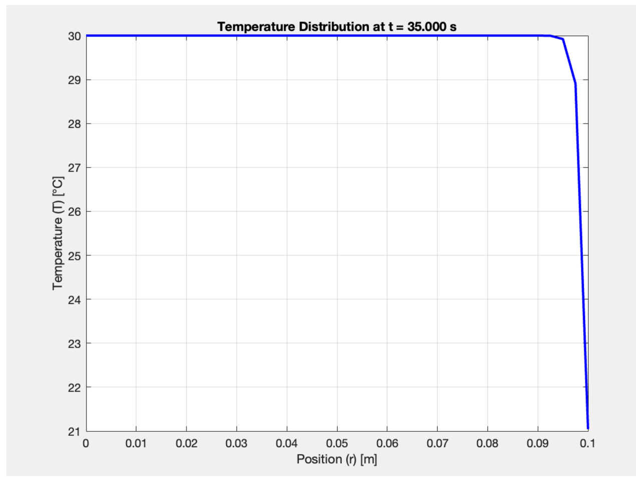
$q_{n \to n+1} = \rho \cdot c_p \cdot dV \cdot \frac{T_n^{p+1} - T_n^p}{dt}$

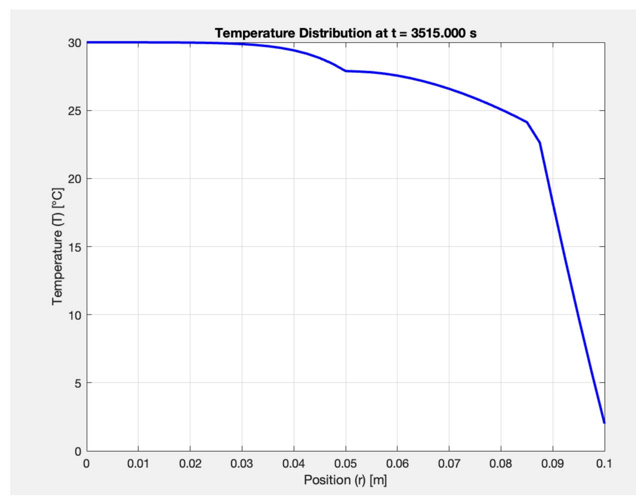
$\frac{k \cdot 4\pi (\frac{dr_n}{2})^2}{dr}(T_{n+1}^p - T_n^p) = \rho \cdot c_p \cdot \frac{4}{3}\pi (\frac{dr}{2})^3 \cdot \frac{T_n^{p+1} - T_n^p}{dt}$

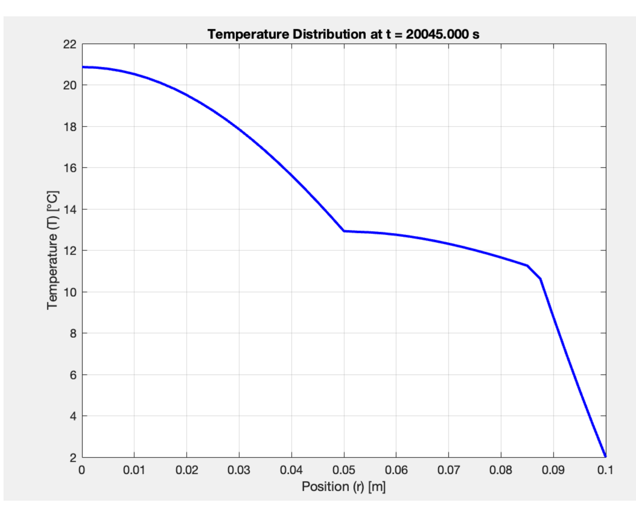
$T_n^{p+1} = T_n^p + \frac{k \cdot 3 (\frac{dr_n}{2})^2 dt / dr}{\rho \cdot c_p (\frac{dr_n}{2})^3} \cdot (T_{n+1}^p - T_n^p)$

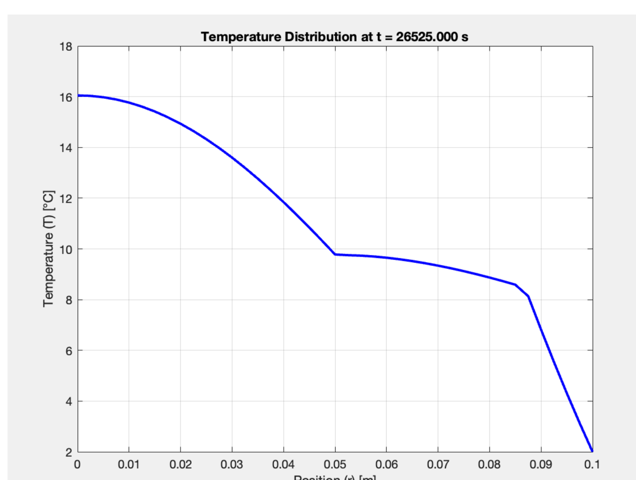
$= T_n^p + \frac{3k\,dt}{\rho c_p (\frac{dr_n}{2}) dr_n} \cdot (T_{n+1}^p - T_n^p)$

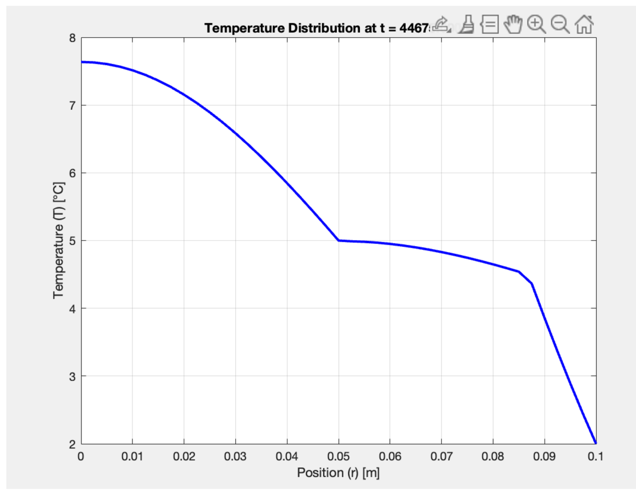
$T_n^{p+1} = T_n^p + \frac{6 \cdot k_n \cdot dt}{\rho \cdot c_p \cdot dr_n^2} \cdot (T_n^p - T_n^p)$





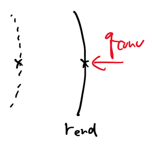






d) Calculate the energy to cool the fruit

Convective node:



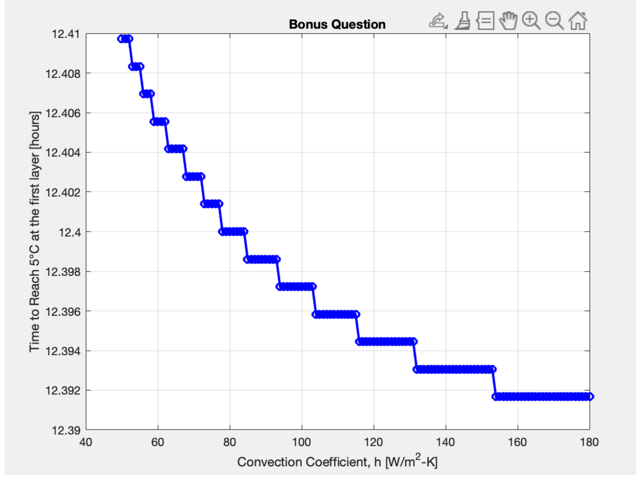
$q_{conv} = h \cdot A_s \cdot \frac{(T_s^p - T_\infty)}{dt}$

$A_s = 4\pi r_{end}^2$

Total energy: 13381.768 J

Bonus question: $h = 50 \to 180\ \frac{W}{m^2 \cdot K}$

h = 50 : 180 ;
h_n = 180 − 50 + 1
} for i = 1 : length(h)

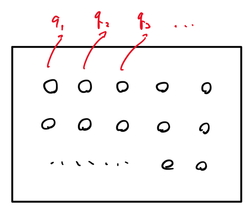


e) My answer from b) estimates that it would take about 8 hours and 30 mins to cool down the fruit under the assumption of mass-weighted average properties

My simulation deals with a property element more locally. Even though I made an assumption that dr is the same throughout the fruit, the simulation gives a profile of 3 different layers distinctively. According to the simulation, with dt = 5, N = 41. It would take much more time than estimate.

My simulation shows 446750 secs = 12.41 hrs to cool it down

f)



$Q = \sum_{i=1}^{500} q_i = 500 \cdot q_1$

$= 500 \cdot 13381.768$

$= 6690884\ J$

It's still 12.41 hours

This is only applicable when each cantaloupe has the same air flow as just one cantaloupe, all the cantaloupes have to have the same initial temperature, and the efficiency of the refrigerator cannot change.

Code for C and D combined

Saturday, December 7, 2024 1:29 AM

```
% This code contains both part C and D
clear all;
% Model constants
radius = [0, 0.05, 0.085, 0.1]; % Include the center and all layer
boundaries
r_tot = radius(end);  % in m
h = 180; % W/m2-K
T_inf = 2; % oC
T_final = 5; % oC
% Material properties
k = [0.03, 0.6, 0.2];
rho = [520, 1000, 1200];
cp = [1470, 4200, 3150];
% Spatial discretization
N = 41;
dr = r_tot/(N-1);
r = [];
% Assign material properties to nodes
k_nodes = zeros(1, N);
rho_nodes = zeros(1, N);
cp_nodes = zeros(1, N);
layer_thickness = diff(radius);
total_thickness = sum(layer_thickness);
nodes_per_layer = round(N * layer_thickness / total_thickness);
% apply material property to each node
index = 1;
for layer = 1:length(nodes_per_layer)

    count = nodes_per_layer(layer);
    % Generate positions for nodes in this layer
    if layer == 1
        r_layer = linspace(radius(layer), radius(layer + 1), count);
% First layer includes center
    else
        r_layer = linspace(radius(layer), radius(layer + 1), count +
1); % Include boundary so +1
        r_layer = r_layer(2:end); % Exclude duplicate boundary from
previous layer
    end
    % Append node positions to r=[]
    r = [r, r_layer]; % Append radial positions
    % Assign material properties for this layer
    k_nodes(index:index + count - 1) = k(layer);
    rho_nodes(index:index + count - 1) = rho(layer);
    cp_nodes(index:index + count - 1) = cp(layer);
    % Update index for the next layer
    index = index + count;
end
r_len = length(r);
% Time discretization
dt = 5; % this is kind of arbiturary becasue the calculated value did
not converge
t_sim = 360000;
N_time = t_sim / dt + 1;
t = 0:dt:t_sim;
% Initial conditions
T_ini = 30; % oC
T_old = ones(r_len, 1) * T_ini;
T_new = T_old;
T_hist = zeros(r_len, N_time);
Q_conv = 0;    % at start
A_outer = 4 * pi * r(end)^2; % Cantalopes' surface area
% Time loop
for i = 1:N_time - 1
    T_old = T_new;

    % Center node
        T_new(1) = T_old(1) + (T_old(2) - T_old(1)) * (6 * k_nodes(1)
* dt)/(rho_nodes(1) * cp_nodes(1) * dr^2);
    % Interior nodes
    for j = 2:N-1

        r_n = (j-1) * dr; % nodal position
        r_minus = r_n - dr / 2;
        r_plus = r_n + dr / 2;

        T_new(j) = T_old(j) + ((T_old(j-1) - T_old(j)) *
k_nodes(j-1)*r_minus^2 / dr + (T_old(j+1) - T_old(j)) * k_nodes(j+1)
* r_plus^2 / dr ) / ((rho_nodes(j)*cp_nodes(j)*(r_n^3-r_minus^3) +
rho_nodes(j)*cp_nodes(j)*(r_plus^3-r_n^3)) / (3*dt));
    end

    % Convection boundary node
    r_outer = r(end);
    r_inner = r(end) - dr / 2;
        T_new(N) = T_old(N) + (T_old(N-1) - T_old(N)) * 3 *
k_nodes(N) * (r_outer - r_inner)^2 * dt / (dr * rho_nodes(N) *
cp_nodes(N) * (r_outer^3 - (r_outer - dr/2)^3)) + (T_inf - T_old(N))
* 3 * h * r_outer^2 * dt / (rho_nodes(N) * cp_nodes(N) * (r_outer^3 -
(r_outer - dr/2)^3));

    % Compute convection energy at the boundary
    dQ_conv = h * A_outer * (T_new(N) - T_inf) * dt; % Convection
heat for each time stamp
    Q_conv = Q_conv + dQ_conv; % Accumulate total convection energy
    % Update history
    T_hist(:, i) = T_new;

    % Plot
    plot(r, T_new, 'b-', 'LineWidth', 2);
    xlabel('Position (r) [m]');
    ylabel('Temperature (T) [°C]');
    title(sprintf('Temperature Distribution at t = %.3f s', i * dt));
    grid on;
    pause(0.01);
    % stopping condition
    if T_new(nodes_per_layer(1)) <= 5
        fprintf('Stopping simulation at time = %.2f seconds\n', i *
dt);
        fprintf('Stopping simulation at time = %.2f hours\n', i * dt
/ 3600);
        fprintf('Temperature at r = 0.05m = %.3f oC\n',
T_new(nodes_per_layer(1)));
        break;
    end
end
% Total energy required
Q_total = Q_conv;
% Display results
fprintf('Total energy required by the refrigeration system: %.3f J
\n', Q_total);
```

Code for Bonus Question

Saturday, December 7, 2024 8:30 PM

```
% This code is for the Bonus Question
clear all;
% Model constants
radius = [0, 0.05, 0.085, 0.1]; % Include the center and all layer
boundaries
r_tot = radius(end);  % in m
T_inf = 2; % oC
T_final = 5; % oC
% Material properties
k = [0.03, 0.6, 0.2];
rho = [520, 1000, 1200];
cp = [1470, 4200, 3150];
% Spatial discretization
N = 41;
dr = r_tot/(N-1);
r = [];
% Assign material properties to nodes
k_nodes = zeros(1, N);
rho_nodes = zeros(1, N);
cp_nodes = zeros(1, N);
layer_thickness = diff(radius);
total_thickness = sum(layer_thickness);
nodes_per_layer = round(N * layer_thickness / total_thickness);
% Apply material property to each node
index = 1;
for layer = 1:length(nodes_per_layer)
    count = nodes_per_layer(layer);
    % Generate positions for nodes in this layer
    if layer == 1
        r_layer = linspace(radius(layer), radius(layer + 1), count);
% First layer includes center
    else
        r_layer = linspace(radius(layer), radius(layer + 1), count +
1); % Include boundary so +1
        r_layer = r_layer(2:end); % Exclude duplicate boundary from
previous layer
    end
    % Append node positions to r=[]
    r = [r, r_layer]; % Append radial positions
    % Assign material properties for this layer
    k_nodes(index:index + count - 1) = k(layer);
    rho_nodes(index:index + count - 1) = rho(layer);
    cp_nodes(index:index + count - 1) = cp(layer);
    % Update index for the next layer
    index = index + count;
end
r_len = length(r);
% Time discretization
dt = 5; % This is arbitrary because the calculated value did not
converge
t_sim = 360000;
N_time = t_sim / dt + 1;
% Initial conditions
T_ini = 30; % oC
% Vary h from 50 to 180
h = 50:1:180; % Convection coefficients to test by the increment of 1
oC
time_to_cool = zeros(size(h)); % Array to store times for the graph
for h_index = 1:length(h)
    h_now = h(h_index); % Current convection coefficient
    T_old = ones(r_len, 1) * T_ini;
    T_new = T_old;
    Q_conv = 0;    % Reset convection energy
    A_outer = 4 * pi * r(end)^2; % Surface area of the cantaloupe
    % Time loop
    for i = 1:N_time - 1
        T_old = T_new;

        % Center node
        T_new(1) = T_old(1) + (T_old(2) - T_old(1)) * (6 * k_nodes(1)
* dt)/(rho_nodes(1) * cp_nodes(1) * dr^2);
        % Interior nodes
        for j = 2:N-1
            r_n = (j-1) * dr; % nodal position
            r_minus = r_n - dr / 2;
            r_plus = r_n + dr / 2;
            T_new(j) = T_old(j) + ((T_old(j-1) - T_old(j)) *
k_nodes(j-1)*r_minus^2 / dr + (T_old(j+1) - T_old(j)) * k_nodes(j+1)
* r_plus^2 / dr ) / ((rho_nodes(j)*cp_nodes(j)*(r_n^3-r_minus^3) +
rho_nodes(j)*cp_nodes(j)*(r_plus^3-r_n^3)) / (3*dt));
        end
        % Convection boundary node
        r_outer = r(end);
        r_inner = r(end) - dr / 2;
            T_new(N) = T_old(N) + (T_old(N-1) - T_old(N)) * 3 *
k_nodes(N) * (r_outer - r_inner)^2 * dt / (dr * rho_nodes(N) *
cp_nodes(N) * (r_outer^3 - (r_outer - dr/2)^3)) + (T_inf - T_old(N))
* 3 * h_now * r_outer^2 * dt / (rho_nodes(N) * cp_nodes(N) *
(r_outer^3 - (r_outer - dr/2)^3));

        % Check stopping condition
        if T_new(nodes_per_layer(1)) <= T_final
            time_to_cool(h_index) = i * dt; % Record time for this
`h` on the each node from 1 to 131
            break;
        end
    end
end
% Plot results
figure;
plot(h, time_to_cool / 3600, 'b-o', 'LineWidth', 2); % Convert time
to hours
xlabel('Convection Coefficient, h [W/m^2-K]');
ylabel('Time to Reach 5°C at the first layer [hours]');
title('Bonus Question');
grid on;
```