```
% ME 303 - Zhao Pan
% Programmers: Shaan B, Zubair H, Mirza M, Dharmik R, Milind K
% Date: 29th March, 2025

clc; % Clear the command window

% Physical egg properties
alpha = 1.26e-7; % Thermal diffusivity of the egg (in m^2/s)
T_init = 4.2; % Initial uniform temp inside the egg (in °C)
T_surface = 100; % Constant temperature at egg surface in (°C)
T_cook = 80; % Minimum internal temperature to be considered cooked (in °C)
cook_time_hold = 10; % Time entire egg must stay at or above T_cook (in s)

% Egg radii (in m). Using a struct array to store egg's name and radius
eggs(1).name = 'Quail';
eggs(1).Radius = 0.0155;
eggs(2).name = 'Chicken';
eggs(2).Radius = 0.02334;
eggs(3).name = 'Ostrich';
eggs(3).Radius = 0.1397;

% Simulation parameters
Nr = 100; % Number of spatial divisions (i.e., radial points in the vector)
max_time = 36000; % Simulate for up to 10 hours (in s)

% Main lööp that runs once for each egg
for e = 1:length(eggs)
    R = eggs(e).Radius; % Get current egg radius (scalar)
    delta_r = R / Nr; % Compute grid spacing based on egg radius
    r = linspace(0, R, Nr+1); % Evenly spaced radii from 0 to R
    delta_t = 0.05 * delta_r^2 / alpha; % Time step based on FTCS stability
    Nt = ceil(max_time / delta_t); % Number of time steps
    time = (0:Nt)*delta_t; % Time vector

    % Initialize temp matrix, where rows = radii, columns = time steps
    T = ones(Nr+1, Nt+1) * T_init; % Fill with uniform initial temperature
    T(end, :) = T_surface; % Apply surface boundary condition (100 °C)

    % Bröther, may I have the lööp that runs FTCS scheme
    for n = 1:Nt
        for i = 2:Nr % Assuming derivation of spherical FTCS is correct
            ri = r(i); % radius at current point
            % Finite difference of second derivative wrt radius
            d2T = (T(i+1,n) - 2*T(i,n) + T(i-1,n)) / delta_r^2;
            % Finite difference of first derivative wrt radius
            dTdr = (T(i+1,n) - T(i-1,n)) / (2*delta_r);
            % Find next point using FTCS derivation in 1b)
            T(i,n+1) = T(i,n) + alpha*delta_t * (d2T + (2/ri)*dTdr);
        end
```

```
        % Symmetry condition at the center (r = 0)
        T(1,n+1) = T(1,n) + 6*alpha*delta_t*(T(2,n) - T(1,n))/delta_r^2;
        % This suggestion came from an LLM
    end

    % Determine when egg is cooked using built-in MATLAB Search Algorithms
    % Check when all points of the egg are ≥ T_cook
    cooked = all(T(1:end-1,:) >= T_cook);
    % Find the first time index where this happens
    cook_start_idx = find(cooked, 1);

    % Find when it stops being cooked again
    cook_end_idx = find(~cooked(cook_start_idx:end), 1);

    % Decisions
    if isempty(cook_start_idx)
        cook_time = NaN; % Never fully cooked

    elseif isempty(cook_end_idx)
        cook_time = time(end) - time(cook_start_idx); % If stayed cooking

    else
        % Compute total duration where it's fully cooked
        cook_time=time(cook_end_idx+cook_start_idx-2)-time(cook_start_idx);
    end

    % More... decisions
    if cook_time >= cook_time_hold
        % If it stayed cooked for at least 10 seconds, store cook time
        cook_time_final = time(cook_start_idx);

    else
        % Otherwise, don't store
        cook_time_final = NaN;
    end

    eggs(e).cook_time = cook_time_final; % Save the result to the struct

    % Plotting core temperature vs. time
    figure; % open new figure
    plot(time, T(1,:), 'b', 'LineWidth', 1.5);
    hold on; % keep multiple plots
    yline(T_cook, 'r--', 'Cook Threshold');
    xlabel('Time (s)'); ylabel('Core Temp (°C)');
    grid on;

    % 3D surface plot Temp vs Radius & Time
    % Reduce resolution for plotting
    stride = max(1, floor(length(time)/100));
    T_sub = T(:,1:stride:end);
```

```
    [Rgrid, Tgrid] = meshgrid(time(1:stride:end), r);
    figure;
    surf(Rgrid, Tgrid, T_sub, 'EdgeColor', 'none'); % Suface gradient
    xlabel('Time (s)'); ylabel('Radius (m)'); zlabel('Temperature (°C)');
    view(135, 30);
    colorbar;
end

% Summary Table
fprintf('\nEstimated Cooking Times:\n');
fprintf('-------------------------\n');
for e = 1:length(eggs)
    if isnan(eggs(e).cook_time)
        fprintf('%s egg: Did not fully cook.\n', eggs(e).name);
    else
        fprintf('%s egg: Cooked at %.1f seconds (%.1f min)\n', ...
            eggs(e).name, eggs(e).cook_time, eggs(e).cook_time / 60);
    end
end
```