MATLAB Simulations of Multi-Dimensional Transient Heat Transfer

By: Beniam Kumela

Table Of Contents

Problem Statement

The general problem chosen is modeling 2D unsteady state heat transfer. The first scenario simulates transient conduction with convective boundary conditions within a solid, square (1x1 m) plate. The surrounding fluid has a temperature of 100 ºC and the initial temperature of the plate is 200 ºC. A heat transfer coefficient of 60 W/m^{2}C was used for convection model. 3 materials were tested with varying thermal conductivities and diffusivities as shown in Table 1 and a schematic of the problem is illustrated in Figure 1.

Table 1: Summary of thermal conductivities and diffusivities of various materials.

Material	**Thermal diffusivity, α (m^2/s)**	**Thermal conductivity, κ (W/mC)**
Diamond	1.2x10^{-3}	2200
Polyvinylchloride Plastic (PVC)	8x10^{-8}	0.19
Copper	1x10^{-4}	386

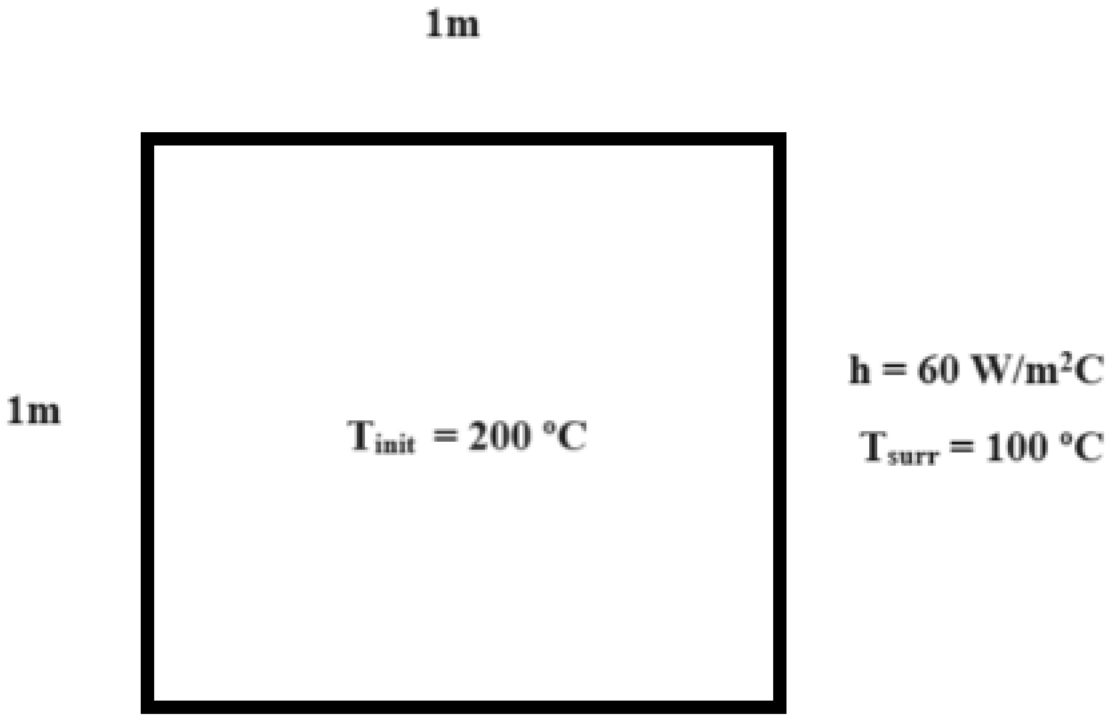


Figure 1: Schematic of conditions described in Scenario 1.

The second scenario is a simpler conduction problem where a Dirichlet boundary condition is used. This just means that the boundaries are maintained at a specified temperature, in this case

100 ºC, with the same geometry and material properties as before. Now, however, only thermal diffusivity, α, is needed. A schematic of this problem is illustrated in Figure 2.

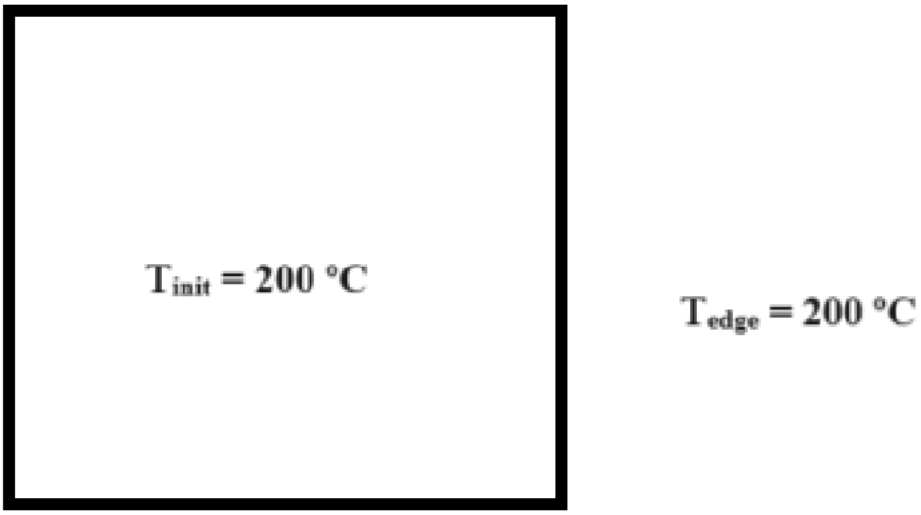


Figure 2: Schematic of conditions described in Scenario 2.

The last scenario looks at a thermocouple junction recording the temperature of a gas stream which is at 200 ºC. As Figure 3 illustrates, the geometry of the exposed thermocouple junction can be approximated by a sphere and thus lumped parameter analysis is viable. A simple schematic of the problem is provided by Figure 4.

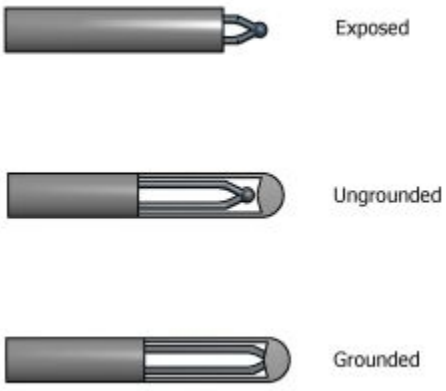


Figure 3: Various types of thermocouple junctions.

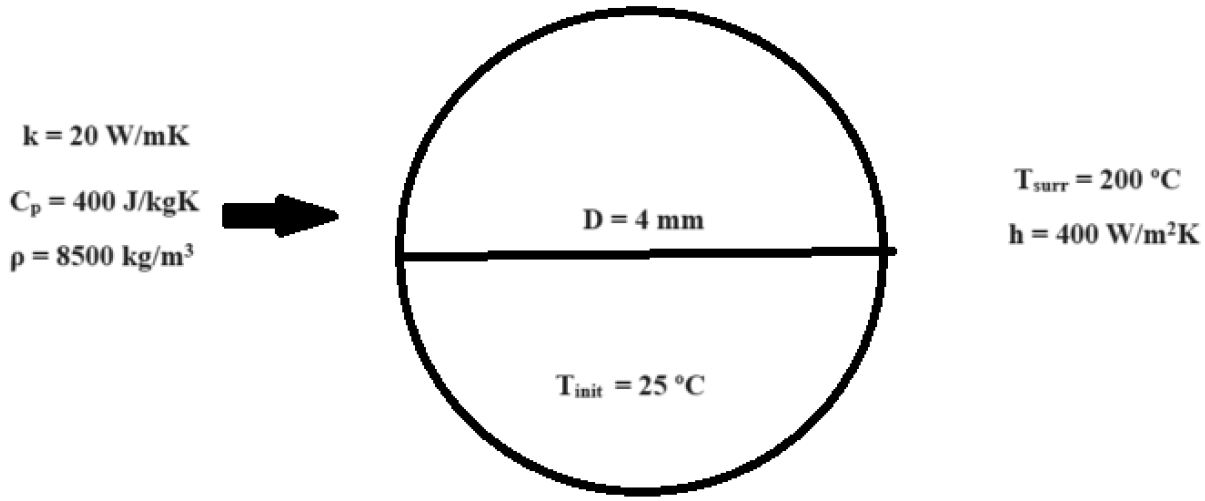


Figure 4: Schematic of Scenario 3.

Summary of Numerical Methods and Calculations

For the first two scenarios, we can derive finite difference equations for unsteady state heat transfer equations directly from the 2D heat equation:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha}\frac{\partial T}{\partial t}$$

where alpha is thermal diffusivity. From forward finite difference equation we know that:

$$f'(a) = \frac{f(a+h) - f(a)}{h}$$

Moving forward we will define “i” as the x-direction, “j” as the y-direction”, and “n” as a certain time point. Coefficients will denote the variable that is being kept constant. The square plate will be subdivided into “nodes” which have a certain temperature at a certain instance in time. This model is illustrated in Figure 5 below.

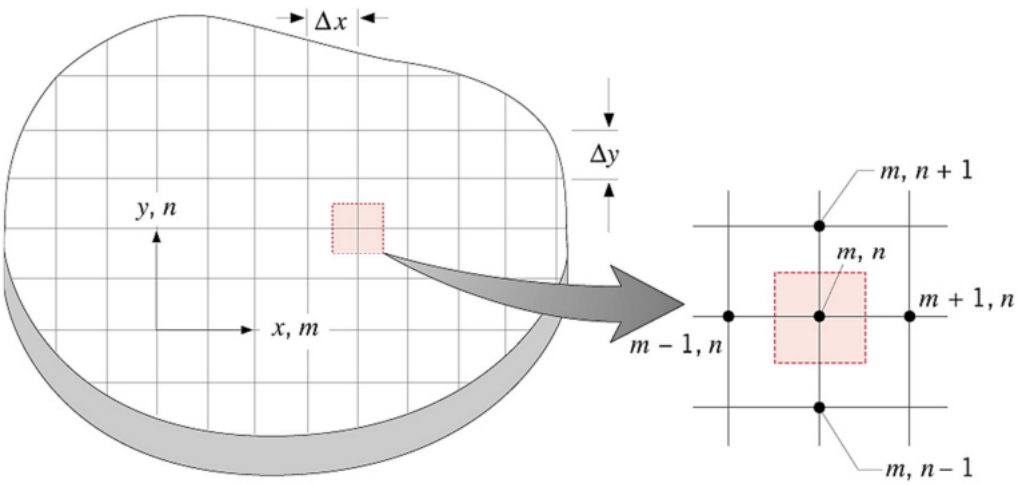


Figure 5: Visualization of space-discretization of an unknown geometry. Note that we defined m=i and n=j in our example.

Thus, we can evaluate the time dependent portion of the equation as:

$$\frac{\partial T}{\partial t} = \frac{T^{ij}(n+1) - T^{ij}(n)}{\Delta t}$$

From the central finite difference method for second derivatives, we know that:

$$f''(a) = \frac{f(a+h) - 2f(a) + f(a-h)}{h^2}$$

From the central finite difference equation for second derivatives, we can evaluate the space dependent portions of the equation as:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T^n(i+1,j) - 2 * T^n(i,j) + T^n(i-1,j)}{\Delta x^2}$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{T^n(i,j+1) - 2 * T^n(i,j) + T^n(i,j-1)}{\Delta y^2}$$

Furthermore, we can take the assumption that dx and dy are equal and plug-in/rearrange the terms from the original heat equation to get the following governing finite difference equation:

$$T^{ij}(n+1) = T^{ij}(n) + \frac{\alpha \Delta t}{\Delta x^2}(T^n(i+1,j) + T^n(i,j+1) + T^n(i-1,j) + T^n(i,j-1) - 4T^n(i,j))$$

Physically, we can say that for each of these “nodes”, the future temperature is equal to the last temperature plus a weighted average of neighboring temperatures. This way as time goes on the temperature of the material will become uniform and average out. The rate of this temperature averaging is determined by that coefficient term which is simplified to “d” (a dimensionless diffusion number) in the MATLAB code.

The last thing that we need to do for the second scenario is hold the outer nodes at 100 ºC. This is done in the MATLAB code by reestablishing this condition after each time iteration.

For the first scenario we will keep the derived finite difference equation but do something different for the boundary conditions. We know the convection boundary conditions which means that we now need to use the equation:

$$-k\frac{dT}{dx} = h(T - T_{surr})$$

Using the forward finite difference equation and rearranging, we can evaluate this expression as:

$$T(n+1) = T(n) + \frac{h\Delta x}{k\Delta t}(T_{surr} - T(n))$$

This governing boundary equation will be applied to the outer border nodes instead of holding the temperature constant as before after each time iteration.

We can determine numerical stability using Von-Neumann stability analysis which determines that the coefficient we defined as d must be less than or equal to 0.25. Sample calculations for this work is shown below:

$$\Delta t \leq \frac{1}{4\alpha\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)} \rightarrow \Delta x \approx \Delta y \rightarrow \Delta t \leq \frac{\Delta x^2}{4\alpha} \qquad \frac{\Delta t\,\alpha}{\Delta x^2} \leq \frac{1}{4}$$

$$d \leq \frac{1}{4}$$

For the final scenario, the geometry of the problem simplifies to a lumped parameter analysis problem which states that for

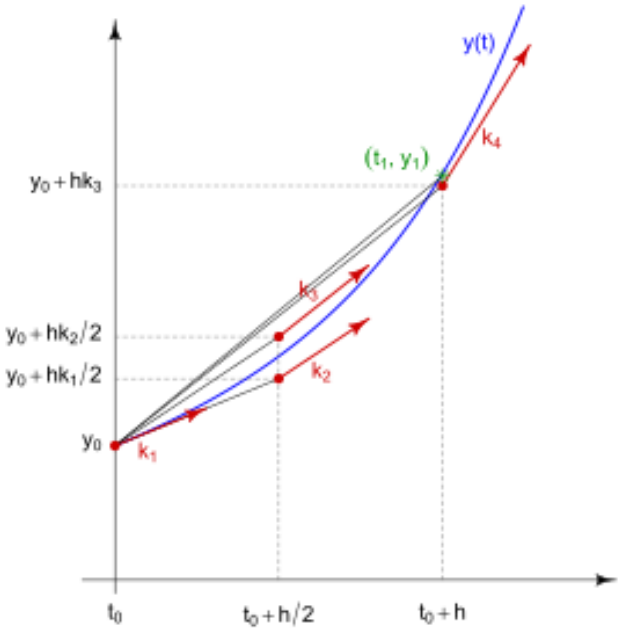
$$Bi = \frac{hV}{Ak} < 0.1$$

where V is volume, A is surface area, k is thermal conductivity, and h is the heat transfer coefficient. Biot's number calculation for the stated conditions is calculated to be 0.08 which is less than 0.1. Thus, the following heat balance expression which neglects the spatial temperature dependence of heat diffusion can be used for heat transfer analysis:

$$\rho C_p V \frac{dT}{dt} = -hA(T - T_{surr})$$

We can simply use the 4th order Runge-Kutta method to approximate this solution as shown below (average of 4 weighted estimates of slope):

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right), \\ k_4 &= f(t_n + h, y_n + hk_3)\,. \end{aligned}$$



The analytical solution is quite simple to derive via the separation of variables method for differential equations, given by the following expression:

$$T = T_{surr} - (T_{surr} - T_0)e^{\frac{-hA}{\rho C_p V}t}$$

Results

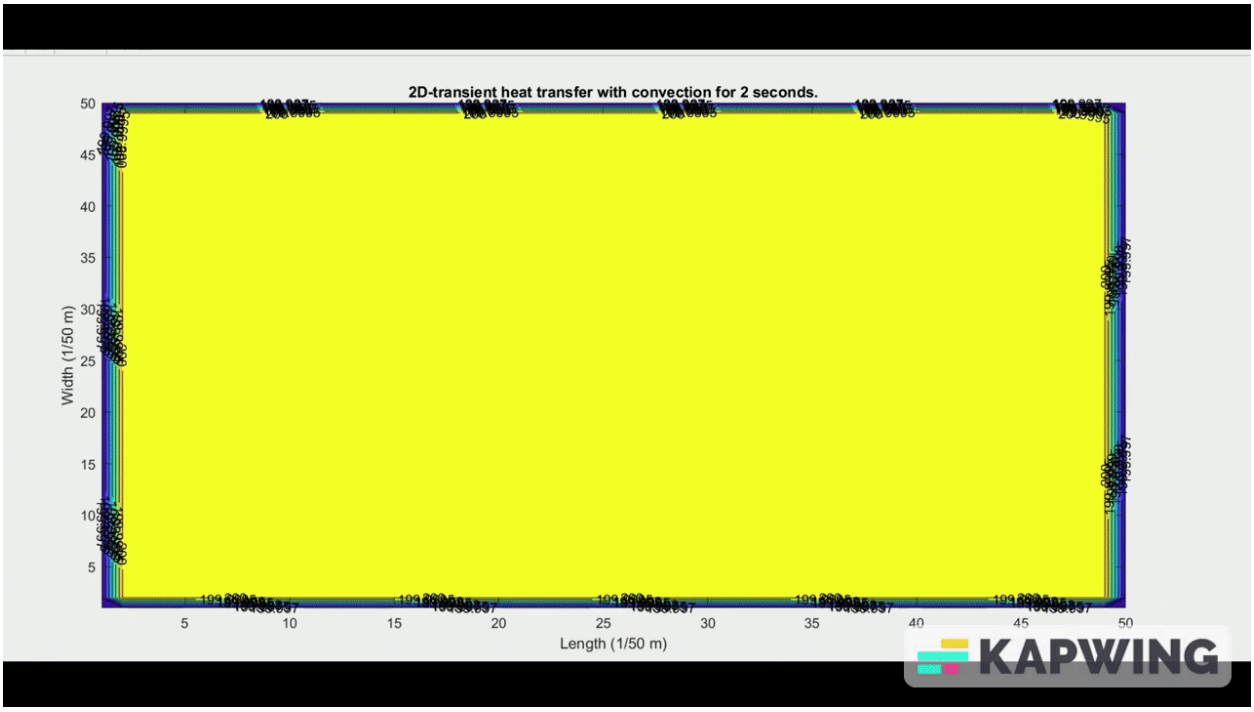


Figure 6: Animated contour plot for copper plate material, convection boundary conditions (Scenario 1).

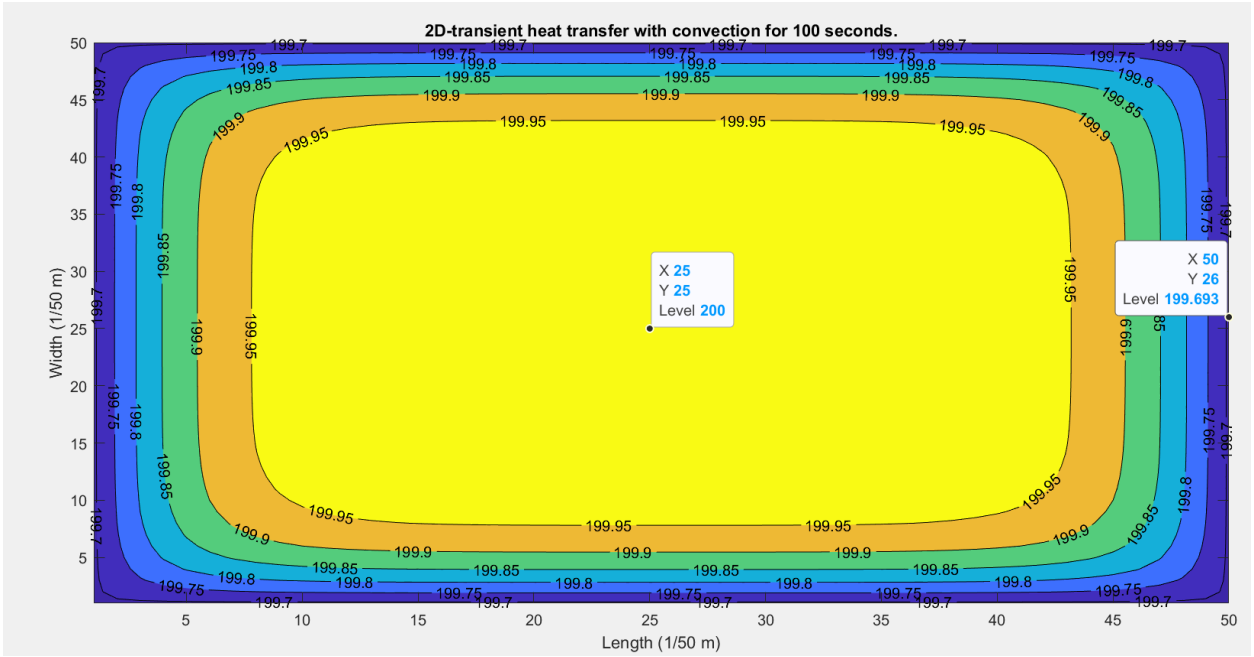


Figure 7: Temperature profile for copper plate, convection boundary conditions, after 100 seconds.

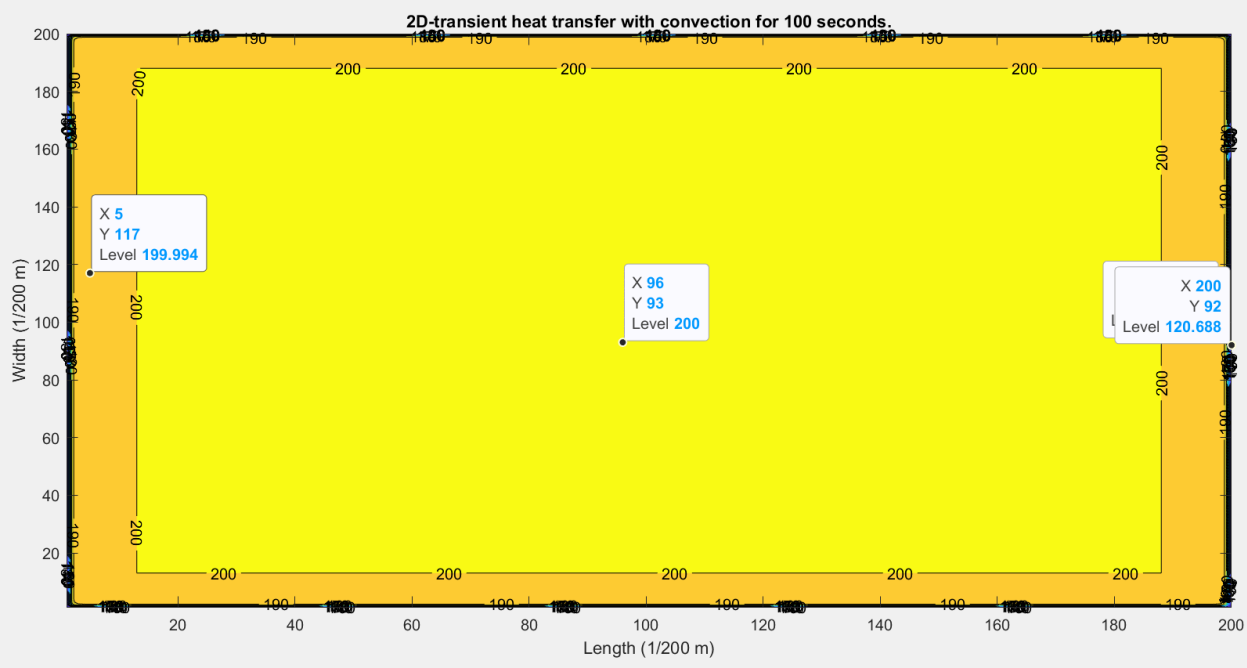


Figure 8: Temperature profile for plastic plate, convection boundary conditions, after 100 seconds.

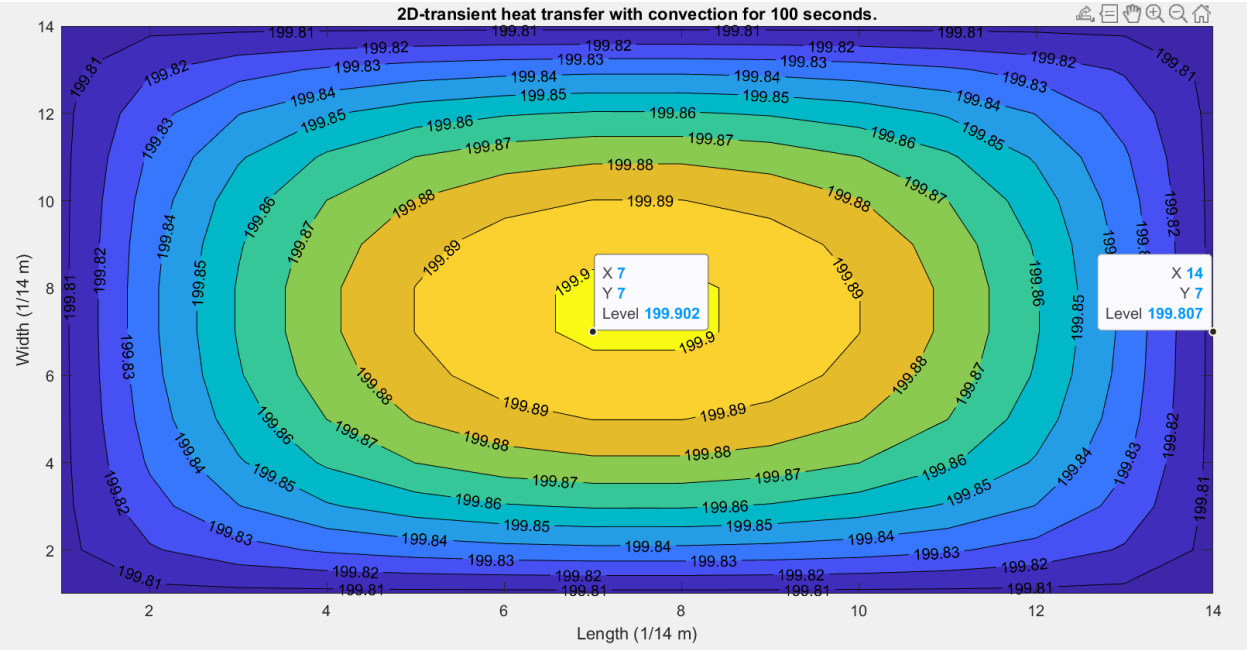


Figure 9: Temperature profile for diamond plate, convection boundary conditions, after 100 seconds.

The numerical solution models the physical phenomena quite well as shown by Figure 6. Over time, the overall temperature profile equilibrates with the outer edges being the coolest due to convection from the cooler fluid flowing across the hot plate. The effect of different materials is quite evident as well. Note how even the temperature profile is for the diamond plate after 10

seconds. The difference between the inner most and outer most point is around 0.1 ºC as shown in Figure 7. The copper plate has a difference of around 0.5 ºC and the plastic plate has a difference of around 80 ºC as shown in Figures 8 and 9. This makes sense given that this is also the order of decreasing thermal conductivities and diffusivities.

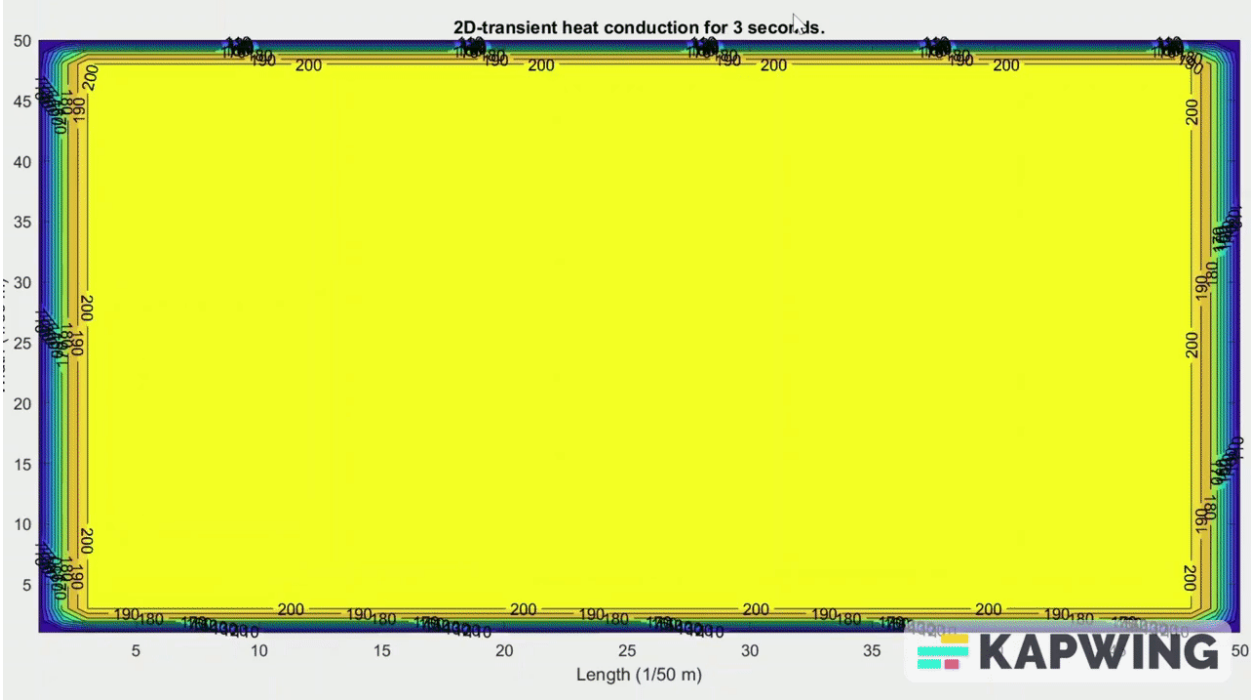


Figure 10: Animated contour plot for copper plate material, constant temperature boundary conditions (Scenario 2).

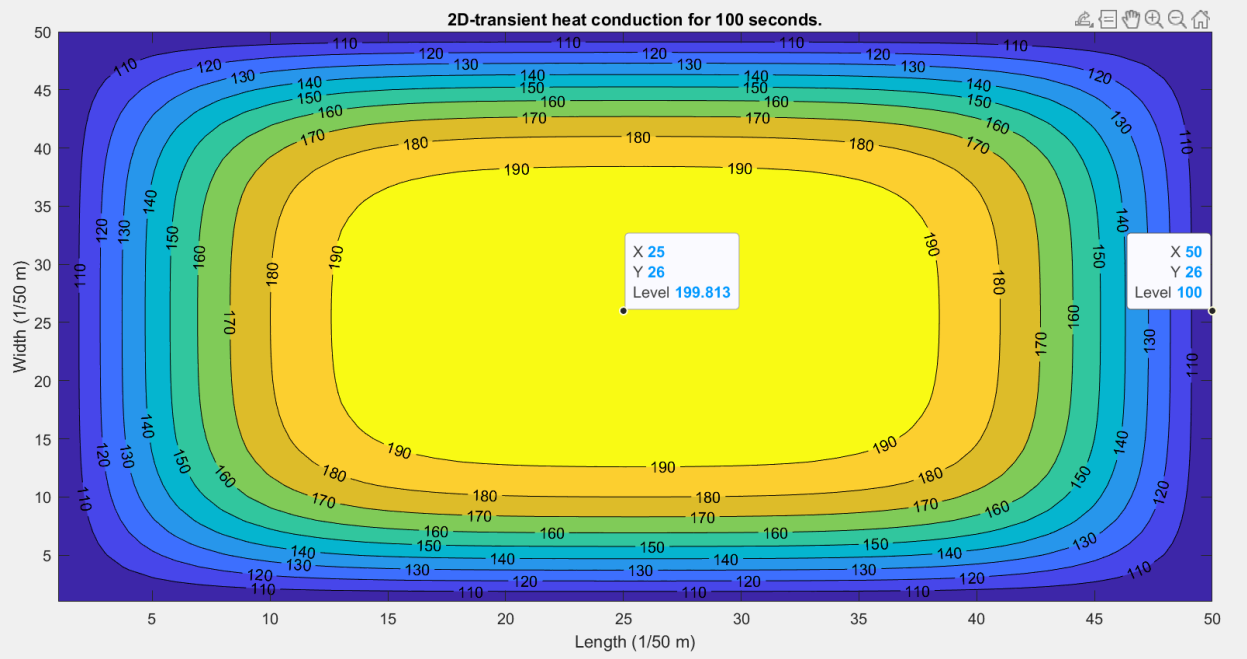


Figure 11: Temperature profile for copper plate, constant temperature boundary conditions, after 100 seconds.

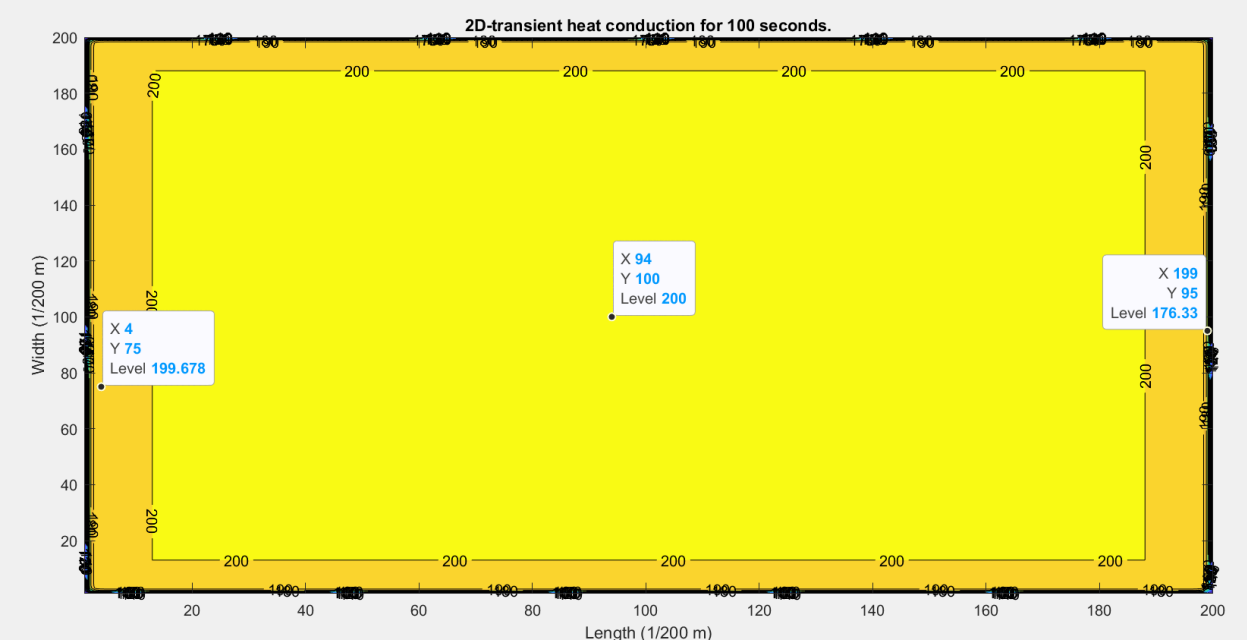


Figure 12: Temperature profile for plastic plate, constant temperature boundary conditions, after 100 seconds.

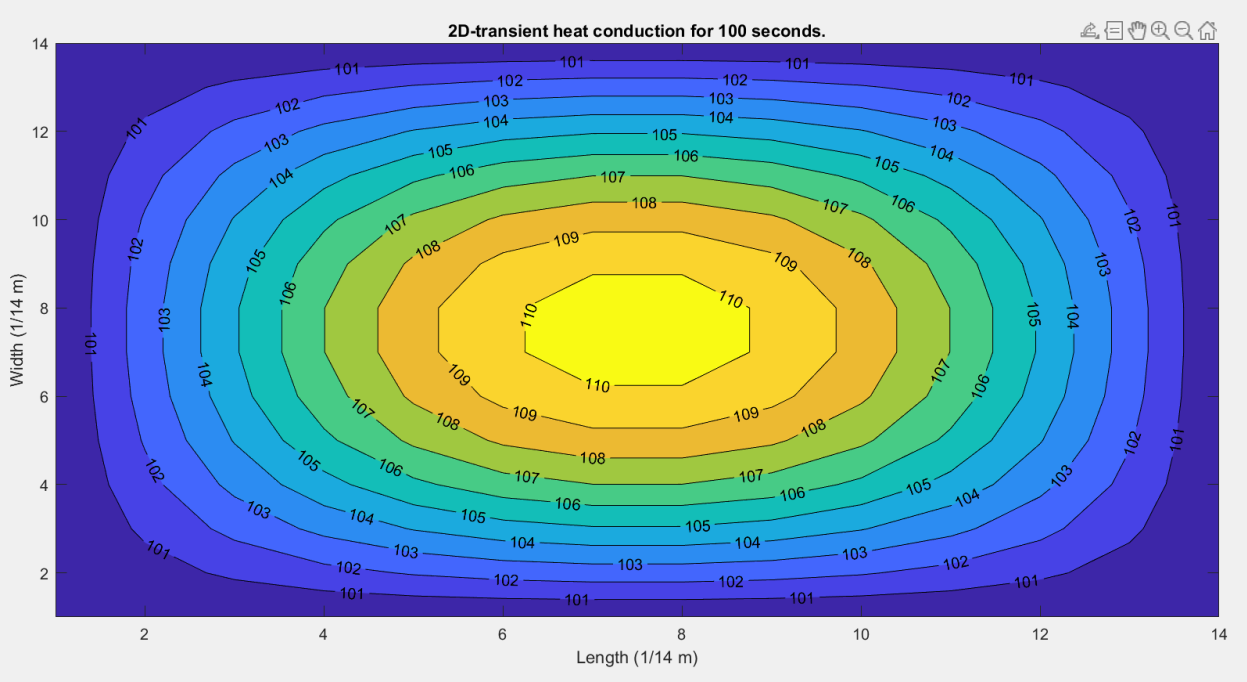


Figure 13: Temperature profile for diamond plate, constant temperature boundary conditions, after 100 seconds.

A similar conclusion can be made for constant temperature boundary condition. The general trend that is followed is the higher the thermal diffusivity of the material, the lower the temperature gradient across the plate. The animated contour plot also provides the same physical picture as before, the plate equilibrates its temperature over time with the outer edges maintaining their constant temperature.

Scenarios 1 and 2 provide basic models and theory that goes into modelling unsteady state heat transfer in complex geometries such as the heat exchanger, a commonplace in chemical engineering plants and applications.

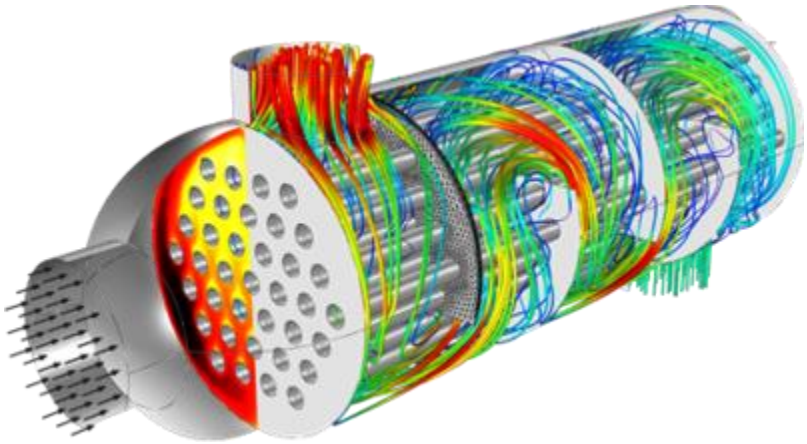


Figure 14: Modelling of Shell-and-Tube Heat Exchanger heat flow in COMSOL, an industry-standard transport phenomenon modelling software.

In the real-world, complex geometries are often modelled in 3D computer aided design software. The finite element method is then used to solve such heat transfer problems. First these geometries are then converted into meshes (kind of like the nodes used in Scenario's 1 and 2) as shown in Figure 15. This results in a system of algebraic equations with boundary conditions. Solving these equations often requires the use of highly optimized sparse matrix solvers and algorithms.

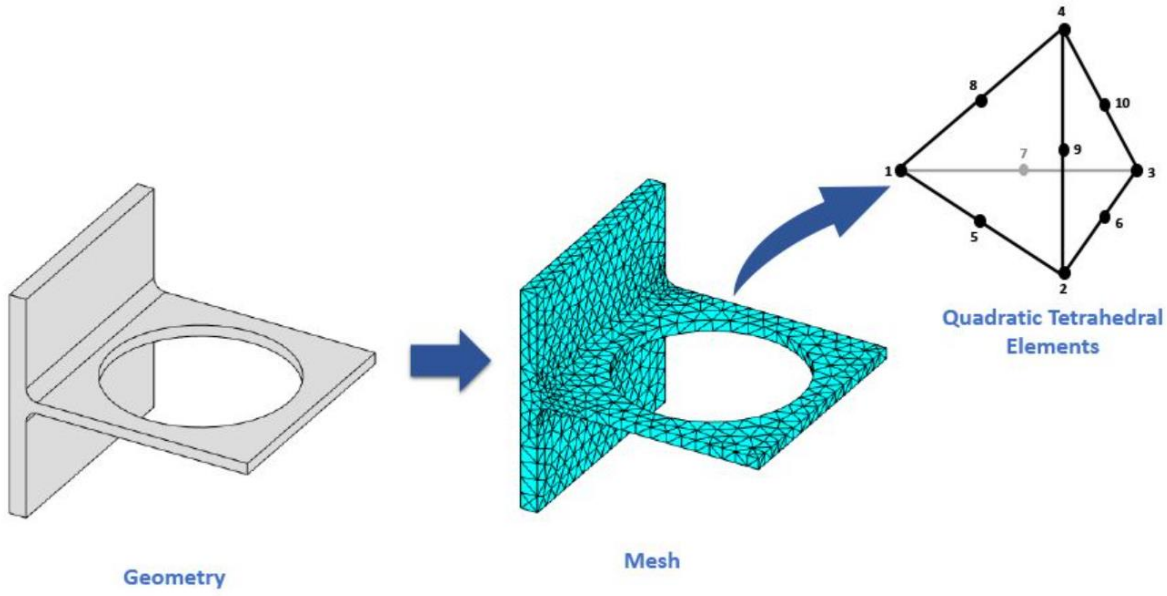


Figure 15: Mesh creation for finite element method.

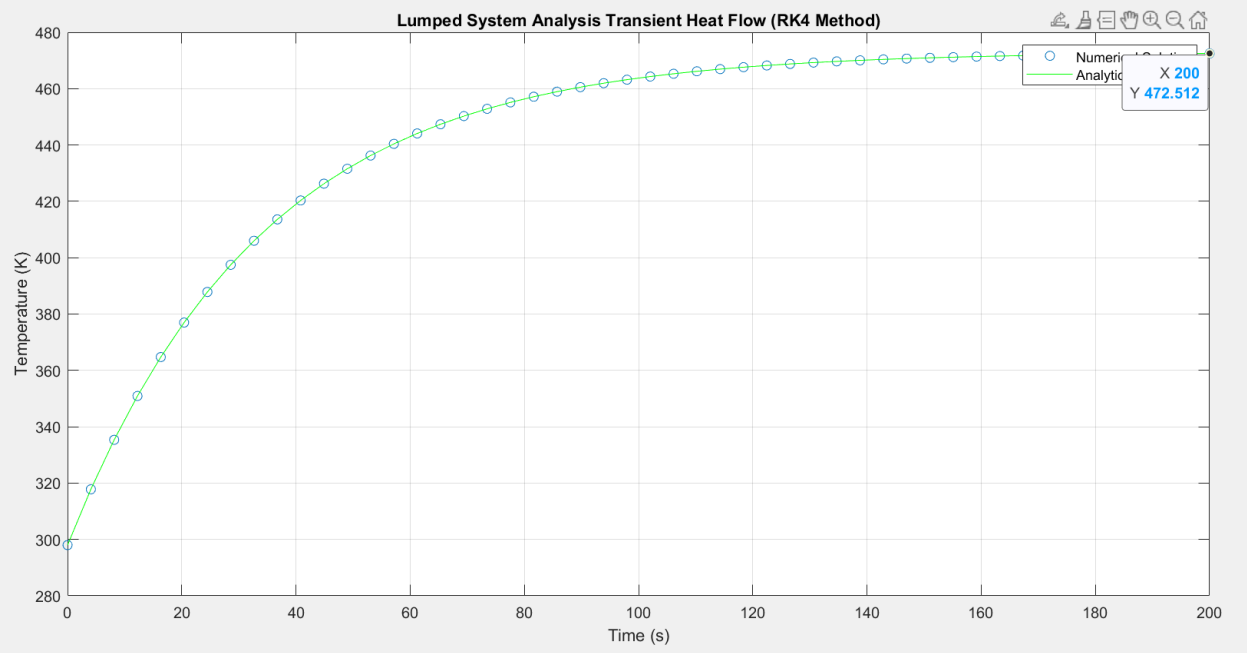


Figure 16: Lumped system analysis treatment for geometry given in Scenario 3 with numerical approximation (RK4) compared with analytical solution over 200 seconds.

For the last scenario it's evident that over time the object reaches a steady state temperature equal to that of the surrounding gas stream (473 K or 200 ºC). It reaches this temperature in approximately 200 seconds as shown in Figure 16. This is approximately the amount of time that the thermocouple would have to be in the gas stream to get an accurate temperature reading.

In this report, several different scenarios regarding heat transfer in the chemical engineering context were explored. The various and extensive numerical methods utilized provided accurate physical pictures of the problems being described. I could have used higher order Runge-Kutta approximations to get more accurate predictions about the last scenario but RK4 was fairly accurate for what was needed. Maybe some implicit, direct, and linear algebra-based schemes could have been used for Scenarios 1 and 2 to reduce computation time needed for the iterative finite difference method utilized. This also reduces the restriction required for stability so smaller time steps or grid sizes could be used. A slightly more accurate model for scenario 1 would have included forced convection and temperature dependent thermal conductivity.

MATLAB Code (also in GitHub if prefer to read there)

Scenario 1 – Convection Boundary Conditions

convPlateFD.m – function which solves using finite difference method

```
function [T_new] = convPlateFD(a, t, nt, length, T_inf, T_init, N, h, k)
    % [T] = convPlateFD(a, t, nt, length, T_inf, T_init, N, h, k)
    %
    % This function solves a PDE boundary value problem for heat
    % conduction on a square plate of form:
    %
    % d^T/dx^2 + d^2T/dy^2 = (1/a)*(dT/dt)
    % on the domain 0 <= x <= 1, 0 <= y <= 1
    %
    % with boundary conditions:
    %
    % h(T_inf - T) = -k(dT/dx) - Convection
    % T(x, y, t=0) = T_init
    %
    % using the Forward Time-Centered Scheme finite difference method.
    %
    % Inputs:
    % a = thermal diffusivity (m^2/s)
    % t = total time (s)
    % nt = number of time steps
    % length = length of square plate side (m)
    % T_inf = surrounding fluid boundary temperature (C)
    % T_init = initial temperature (C)
    % N = spatial grid size
    % h = heat transfer coefficient (W/m^2 K)
    % k = thermal conductivity (W/m K)
    %
    % Outputs:
    % T = NxN array
    %
    % Written by Beniam Kumela 10/31/2023
```

```
% Error checking
if (a <= 0)
    error('condPlateFD: a must be greater than 0.')
end

if (t <= 0)
    error('condPlateFD: t must be greater than 0.')
end

if (nt <= 0)
    error('condPlateFD: nt must be greater than 0.')
end
if (length <= 0)
    error('condPlateFD: length must be greater than 0.')
end

if (T_inf == T_init)
    error('condPlateFD: T_inf cannot be equal to T_init.')
end

if (round(N) ~= N || (N < 3))
    error('condPlateFD: N must be positive integer greater than 3.')
end

% Initialize time step, step length, and dimensionless
% diffusion numbers.
dt = (t/nt);
dx = (length/N);
d = (a*dt)/(dx^2);
Bi = dx / (k*nt); % Step size adjusted Biot's number

% Initialize temperature grids
T_old = ones(N, N);
T_new = T_old;

% Initial temperature distribution of plate
T_old(:, :, 1) = T_init;
T_new(:, :, 1) = T_init;

% FTCS finite difference method algorithm derived from heat equation
for n=1:nt
    for i=2:N-1
        for j=2:N-1
            T_new(i, j, n+1) = T_new(i, j, n) + d * (T_new(i+1, j, n) ...
                + T_new(i, j+1, n) + T_new(i-1, j, n) + T_new(i, j-1, n) ...
                - 4 * T_new(i, j, n));
        end
    end

    % Establish convection boundary conditions derived from eqn above
    T_new(1,:,n+1) = T_new(1,:,n) + ...
        h * (T_inf - T_new(1,:,n)) * Bi;% Top side
    T_new(N,:,n+1) = T_new(N,:,n) + ...
        h * (T_inf - T_new(N,:,n)) * Bi;% Bottom side
```

```
        T_new(:,1,n+1) = T_new(:,1,n) + ...
            h * (T_inf - T_new(:,1,n)) * Bi;% Left side
        T_new(:,N,n+1) = T_new(:,N,n) + ...
            h * (T_inf - T_new(:,N,n)) * Bi;% Right side

        % Swap grids
        T_old = T_new;

    end

    % Check that solution obtained was numerically stable.
    if (d <= 0.25)
        disp(['condPlateFD: Solution converged. D = ', num2str(d), ' and is' ...
            ' numerically stable.'])
    else
        disp(['condPlateFD: Solution did not converge. D = ', num2str(d), ...
            ' needs to be <= 0.25.'])
    end

    % Return final grid
    T_new = T_new;
end
```

test_convPlateFD.m – test script for function

```
% This script tests the function convPlateFD
%
% Define parameters,
% Thermal diffusivities (copper and plastic) obtained from:
% https://www.engineersedge.com/heat_transfer/thermal_diffusivity_table_13953.htm
%
% Diamond (99% 12C) obtained from following paper:
% https://web.stanford.edu/group/fayer/articles/176.pdf
%
% a = thermal diffusivity of material (m^2/s)
% t = total time observing is (s)
% N = length of square grid
% k = thermal conductivity (W/mC)

% Case 1 - copper
% a = 1e-4;
% t = 100;
% N = 25;
% k = 386;

% Case 2 - PVC plastic
% a = 8e-8;
% t = 100000;
% N = 16;
% k = 0.19;

% Case 3 - Diamond
a = 1.2e-3;
t = 10;
```

```
N = 45;
k = 2200;

h = 60; % heat transfer coefficient (W/m^2C)
nt = 100; % number of time steps
length = 1; % length of square plate (m)
T_inf = 100; % temperature of surrounding air (C)
T_init = 200; % inital temperature of square copper plate (C)

% Call function to obtain spatial temperature vs time grid
[T] = convPlateFD(a, t, nt, length, T_inf, T_init, N, h, k);

% Plot contour heat map of spatial temperature time evolution
figure(1)
for i = 2:nt
    contourf(T(:, :, i), 'linecolor', 'non')
    title(['2D-transient heat transfer with convection for ' ...
        , num2str(i*(t/nt)), ' seconds.'])
    xlabel(['Length (1/', num2str(N), ' m)'])
    ylabel(['Width (1/', num2str(N), ' m)'])
    % colorbar
    hold on
    contour(T(:, :, i), 'linecolor', 'k');
    h = clabel(contour(T(:, :, i)), 'LabelSpacing', 100);
    hold off
    drawnow
    pause(0.1)
end
```

Scenario 2 – Boundary Temperature Conditions

condPlateFD.m – function solver using finite difference method

```
function [T_new] = convPlateFD(a, t, nt, length, T_inf, T_init, N, h, k)
    % [T] = convPlateFD(a, t, nt, length, T_inf, T_init, N, h, k)
    %
    % This function solves a PDE boundary value problem for heat
    % conduction on a square plate of form:
    %
    % d^T/dx^2 + d^2T/dy^2 = (1/a)*(dT/dt)
    % on the domain 0 <= x <= 1, 0 <= y <= 1
    %
    % with boundary conditions:
    %
    % h(T_inf - T) = -k(dT/dx) - Convection
    % T(x, y, t=0) = T_init
    %
    % using the Forward Time-Centered Scheme finite difference method.
    %
    % Inputs:
    % a = thermal diffusivity (m^2/s)
    % t = total time (s)
    % nt = number of time steps
    % length = length of square plate side (m)
```

```
% T_inf = surrounding fluid boundary temperature (C)
% T_init = initial temperature (C)
% N = spatial grid size
% h = heat transfer coefficient (W/m^2 K)
% k = thermal conductivity (W/m K)
%
% Outputs:
% T = NxN array
%
% Written by Beniam Kumela 10/31/2023

% Error checking
if (a <= 0)
    error('condPlateFD: a must be greater than 0.')
end

if (t <= 0)
    error('condPlateFD: t must be greater than 0.')
end

if (nt <= 0)
    error('condPlateFD: nt must be greater than 0.')
end
if (length <= 0)
    error('condPlateFD: length must be greater than 0.')
end

if (T_inf == T_init)
    error('condPlateFD: T_inf cannot be equal to T_init.')
end

if (round(N) ~= N || (N < 3))
    error('condPlateFD: N must be positive integer greater than 3.')
end

% Initialize time step, step length, and dimensionless
% diffusion numbers.
dt = (t/nt);
dx = (length/N);
d = (a*dt)/(dx^2);
Bi = dx / (k*nt); % Step size adjusted Biot's number

% Initialize temperature grids
T_old = ones(N, N);
T_new = T_old;

% Initial temperature distribution of plate
T_old(:, :, 1) = T_init;
T_new(:, :, 1) = T_init;

% FTCS finite difference method algorithm derived from heat equation
for n=1:nt
    for i=2:N-1
        for j=2:N-1
            T_new(i, j, n+1) = T_new(i, j, n) + d * (T_new(i+1, j, n) ...
```

```
                    + T_new(i, j+1, n) + T_new(i-1, j, n) + T_new(i, j-1, n) ...
                    - 4 * T_new(i, j, n));
            end
        end

        % Establish convection boundary conditions derived from eqn above
        T_new(1,:,n+1) = T_new(1,:,n) + ...
            h * (T_inf - T_new(1,:,n)) * Bi;% Top side
        T_new(N,:,n+1) = T_new(N,:,n) + ...
            h * (T_inf - T_new(N,:,n)) * Bi;% Bottom side
        T_new(:,1,n+1) = T_new(:,1,n) + ...
            h * (T_inf - T_new(:,1,n)) * Bi;% Left side
        T_new(:,N,n+1) = T_new(:,N,n) + ...
            h * (T_inf - T_new(:,N,n)) * Bi;% Right side

        % Swap grids
        T_old = T_new;

    end

    % Check that solution obtained was numerically stable.
    if (d <= 0.25)
        disp(['condPlateFD: Solution converged. D = ', num2str(d), ' and is' ...
            ' numerically stable.'])
    else
        disp(['condPlateFD: Solution did not converge. D = ', num2str(d), ...
            ' needs to be <= 0.25.'])
    end

    % Return final grid
    T_new = T_new;
end
```

test_condPlateFD.m – test script for function

```
% This script tests the function condPlateFD
%
% Define parameters,
% Thermal diffusivities (copper and plastic) obtained from:
% https://www.engineersedge.com/heat_transfer/thermal_diffusivity_table_13953.htm
%
% Diamond (99% 12C) obtained from following paper:
% https://web.stanford.edu/group/fayer/articles/176.pdf
%
% a = thermal diffusivity of material (m^2/s)
% t = total time observing is (s)
% N = length of square grid

% Case 1 - copper
% a = 1e-4;
% t = 100;
% N = 25;

% Case 2 - PVC plastic
% a = 8e-8;
% t = 100000;
```

```
% N = 16;

%Case 3 - Diamond
a = 1.2e-3;
t = 10;
N = 45;

nt = 100; % number of time steps
length = 1; % length of square plate (m)
T_surr = 100; % temperature of surrounding air (C)
T_init = 200; % inital temperature of square copper plate (C)

% Call function to obtain spatial temperature vs time grid
[T] = condPlateFD(a, t, nt, length, T_surr, T_init, N);

% Plot contour heat map of spatial temperature time evolution
figure(1)
for i = 2:nt
    contourf(T(:, :, i), 'linecolor', 'non')
    title(['2D-transient heat conduction for ', num2str(i*(t/nt)), ' seconds.'])
    xlabel(['Length (1/', num2str(N), ')'])
    ylabel(['Width (1/', num2str(N), ')'])
    %colorbar
    hold on
    contour(T(:, :, i), 'linecolor', 'k');
    h = clabel(contour(T(:, :, i)), 'LabelSpacing', 100);
    hold off
    drawnow
    pause(0.1)
end
```

Scenario 3 – Lumped Parameter Heat Analysis

odeRK4.m – function solver for 4th order Runge-Kutta method

```
function [t, y] = odeRK4(f, tvec, y0)
    % function [t, y] = odeRK4(f, tvec, y0)
    %
    % This function integrates an ODE using the 4th order Runge-Kutta Method.
    %
    % ODE: y' = f(t,y)
    %

    % Initialize variables
    t = tvec;
    N = length(t);
    y = zeros(N, 1);
    dt = t(2) - t(1);
    dt2 = 0.5*dt;
    y(1) = y0;
    maxIter = 100;
    tol = sqrt(eps);

    % Integrate with 4th order Runge-Kutta Method
```

```
    for n=1:N-1
        k1 = f(t(n), y(n));
        k2 = f(t(n) + dt2, y(n) + dt2*k1);
        k3 = f(t(n) + dt2, y(n) + dt2*k2);
        k4 = f(t(n) + dt, y(n) + dt*k3);
        y(n+1) = y(n) + dt*((k1/6) + (k2/3) + (k3/3) + (k4/6));
    end
end
```

lumpCondIVP.m – test script for Scenario 3

```
% This script solves the lumped parameter heat analysis problem of the
% form:
%
% rho*Cp*V*dT/dt = -hA(T-T_surr)
%
% using the RK4 method.
%

% Initialize variables
N = 200;
tvec = linspace(0, 200, N)';
T_init = 25 + 273;
T_surr = 200 + 273;
rho = 8500;
Cp = 400;
k = 20;
h = 400;
d = 0.004;
Bi = h*d/k;

% Check validity of lumped parameter analysis
if (Bi < 0.1)
    disp(['Lumped parameter analysis is valid. Biot Number: ', num2str(Bi)])
else
    disp(['Lumped parameter analysis is not valid. Biot Number: ', num2str(Bi) ...
        , ' It needs to be less than 0.1'])
end

% Integrate with Runge-Kutta
[t, T_numerical] = odeRK4((@(t, T) -h * (T-T_surr)/(rho * Cp * d)) ...
    , tvec, T_init);

% Analytical solution
T_actual = T_surr - (T_surr - T_init) * exp((-h / (rho * Cp * d)) * t);

% Compute error
error = norm(T_actual-T_numerical);

% Plot solution
figure(1)
plot(t, T_numerical, 'o', t, T_actual, 'g')
title('Lumped System Analysis Transient Heat Flow (RK4 Method)')
xlabel('Time (s)')
```

```
ylabel('Temperature (K)')
legend('Numerical Solution', 'Analytical Solution')
grid
```

Bibliography

Bird, R. Byron, and I. Warren Stewart. *Transport Phenomena*. J. Wiley, 2007.

Edge, Engineers. "Thermal Diffusivity Table." *Engineers Edge - Engineering, Design and Manufacturing Solutions*, www.engineersedge.com/heat_transfer/thermal_diffusivity_table_13953.htm. Accessed 20 Nov. 2023.

FDM for Heat Transferring - Visual Slope, visualslope.com/Library/FDM-for-heat-transfering.pdf. Accessed 20 Nov. 2023.

HyperPhysics. "Thermal Conductivity Table." *Thermal Conductivity*, hyperphysics.phy-astr.gsu.edu/hbase/Tables/thrcn.html. Accessed 20 Nov. 2023.

Murthy, Jayathi. *Numerical Methods in Heat, Mass, and Momentum Transfer*, 2002, engineering.purdue.edu/~scalo/menu/teaching/me608/ME608_Notes_Murthy.pdf.

"Plastics - Thermal Conductivity Coefficients." *Engineering ToolBox*, www.engineeringtoolbox.com/thermal-conductivity-plastics-d_1786.html. Accessed 20 Nov. 2023.

"Selecting Your Junction Type." *Temprel Temperature Sensors*, www.temprel.com/learning/selecting-your-junction-type. Accessed 20 Nov. 2023.

"Shell-and-Tube Heat Exchanger." *COMSOL*, www.comsol.com/model/shell-and-tube-heat-exchanger-12685. Accessed 20 Nov. 2023.

Tokmakoff, A., et al. "Thermal diffusivity measurements of natural and isotopically enriched diamond by picosecond infrared transient grating experiments." *Applied Physics A Solids and Surfaces*, vol. 56, no. 1, 1993, pp. 87–90, https://doi.org/10.1007/bf00351908.

"Transient Heat Conduction - Simon Fraser University." *Simon Frasier University*, www.sfu.ca/~mbahrami/ENSC%20388/Notes/Transient%20Heat%20Conduction.pdf. Accessed 20 Nov. 2023.

"Transient Heat Conduction FTCS Method." *Engineering Mechanical Process Calculations*, engineering-stream.com/Power_Point_Links/2D_Heat%20Conduction_Cart_Coordinates_Transient_FTCS_ppt_pdf.html. Accessed 20 Nov. 2023.

"Von Neumann Stability Analysis." *Wikipedia*, Wikimedia Foundation, 28 Dec. 2022, en.wikipedia.org/wiki/Von_Neumann_stability_analysis.

"What Is Finite Element Analysis?" *What Is Finite Element Analysis? - MATLAB & Simulink*, www.mathworks.com/discovery/finite-element-analysis.html. Accessed 20 Nov. 2023.