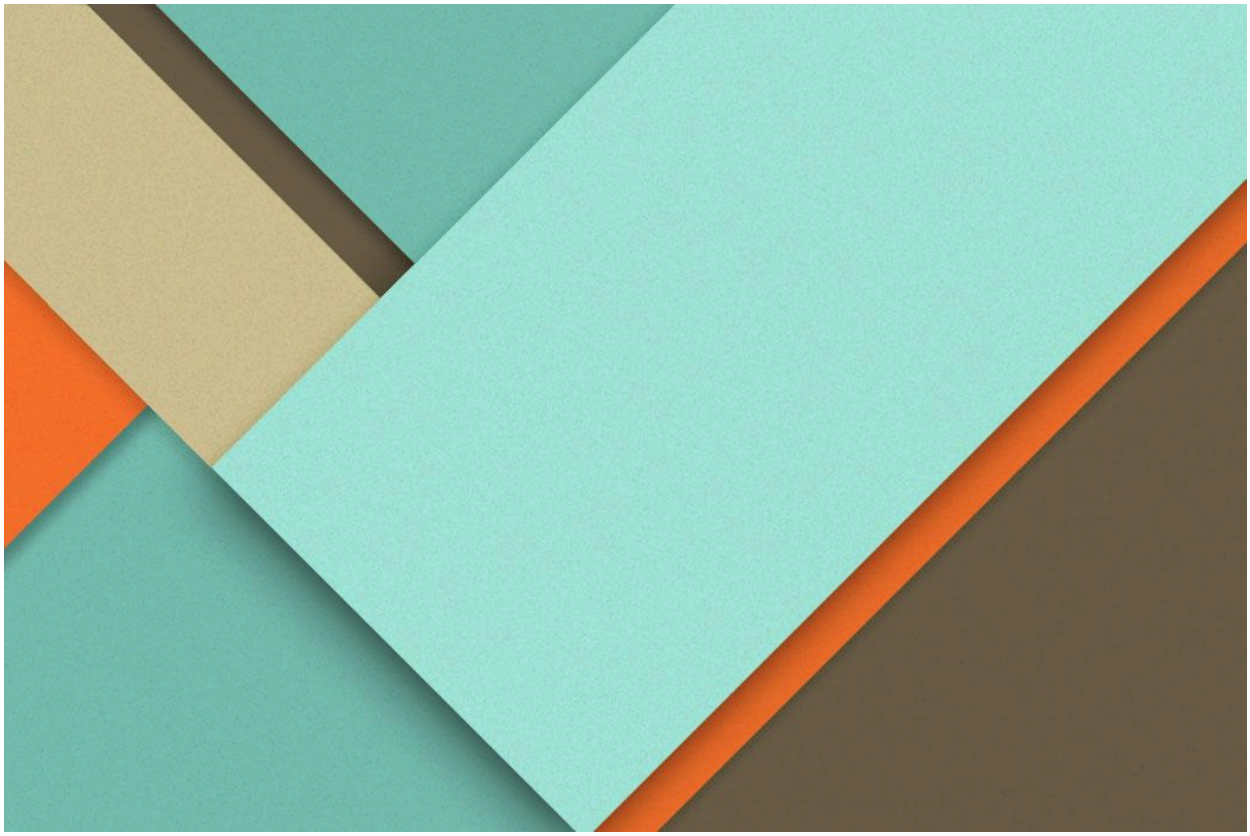


Assignment 1

Question 1

Abhinav Prakash
21CH10001
Course Code: CH620003

Problem Statement

The Fourier heat conduction equation in 1D space is

$$\frac{\partial}{\partial t}(\rho c_p T) = \frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right]$$

where $k(T)$ is the temperature dependent thermal conductivity of SILVER. You can use any available resource to find out the $k(T)$, either regressing (polynomial fit) the temperature versus thermal conductivity data, or using a mathematical relationship. But remember $k(T)$ is not $constant$ and should be considered to be function of T.

Consider the density ρ and specific heat capacity c_p as constant, for SILVER. Please find these values from some resource (e.g. google / perry's handbook / etc.)

The initial and boundary conditions are-

at $t = 0, T = T_0 = 300\ K$ and at $x = 0, T = T_0[1 + \sin(2\pi\omega t)]$

where ω is the frequency of the fluctuation. The temperature at one end of the domain follows a sinusoidal variation. This particular problem is relevant in heating of electric-vehicle charging ports connected to an AC source. You can relate the ω to be frequency of the AC supply.

at $x = L = 1cm, T = T_0$ (constant temperature)

Calculate the temperature profile $T(x = 0.25L, t)$, $T(x = 0.5L, t)$, $T(x = 0.75L, t)$ and $T(x = L, t)$ using (a) Explicit and (b) Implicit finite difference numerical scheme, and (c) Crank Nicholsan method. You need to calculate the temperature profile for at least 5 cycles of ω. For e.g., if $\omega = 1$ Hz, you need to calculate for the final time up to 5s; for $\omega = 50\ Hz$, final time should be computed up to 100 ms.

Note for the explicit scheme, you need to take care of the Von-Neumann stability criterion.

(d) Using the explicit scheme, change the parameter $\lambda = \frac{\alpha \Delta t}{(\Delta x)^2}$ to check the numerical convergence. Show how the solution behaves when λ exceeds the von-Neumann stability criteria.

Compare the explicit results with the implicit scheme, in terms of accuracy of the solution.

(e) Change the number of internal grid points from 10 to 1000 to 10^5, and estimate the computational time needed to solve the tri-diagonal matrix (TDM) in the Implicit scheme using classical matrix inversion techniques (gauss-elimination, gauss-Jordan, gauss-newton, etc.) vis-à-vis the Thomas algorithm.

(f) Plot $T(x = 0, t)$ and length averaged $\bar{T}(t) = \frac{1}{L}\int_0^L T(x, t)\ dx$ with t, for different values of of ω. Is there any phase difference observed between the average temperature response, with respect to $x = 0$.

Determine the critical value of ω, beyond which the temperature variations at the other tip $[T(x = L, t)]$ is minimal. Does this critical value of ω depend on α ? Support your answer with results.

Google Colab Link

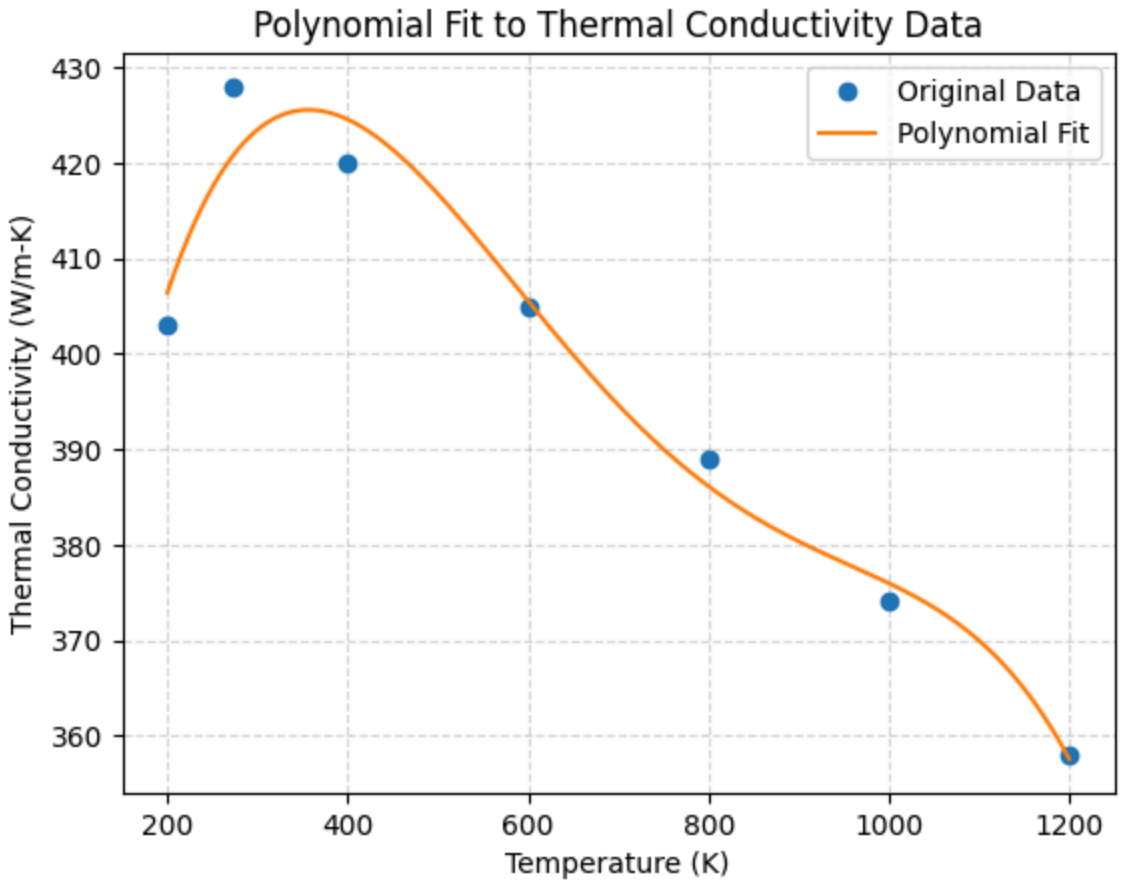
https://colab.research.google.com/drive/1SV4PzDeXjW8GkiGriQHrRm0vPq31fZnB#scrollTo=RpaQ9u76CV3C

Kt

Lets first find the data to be able to apply regression and k(t) as a function of time.

Temperature(K)	Thermal Conductivity(W/mK)
200	403
273	428
400	420
600	405
800	389
1000	374
1200	358

Output



Polynomial Coefficients:

[-7.07404048e-10 2.18162018e-06 -2.37539054e-03 9.90019950e-01 2.87119101e+02]

Here the first 2 terms are almost equal to 0. So we can ignore them. So, we are left with
[-2.37539054e-03 9.90019950e-01 2.87119101e+02]

Physical properties of Silver:

$$\rho = 10500\, kg/m^3$$
$$Cp = 235\, J/KgC$$

Boundary condition :
at x = 0, T = T0[1 + sin(2πωt)]
at x = L = 1 cm, T = T0

Initial condition
at t = 0, T = T0 = 300 K

Explicit Method

Discretized equation to solve for:

$$\rho Cp[T_i^{n=1} - T_i^n]/\Delta = K(T)\,[T^n_{i+1} - 2*T_i^n + T^n_{i-1}]$$

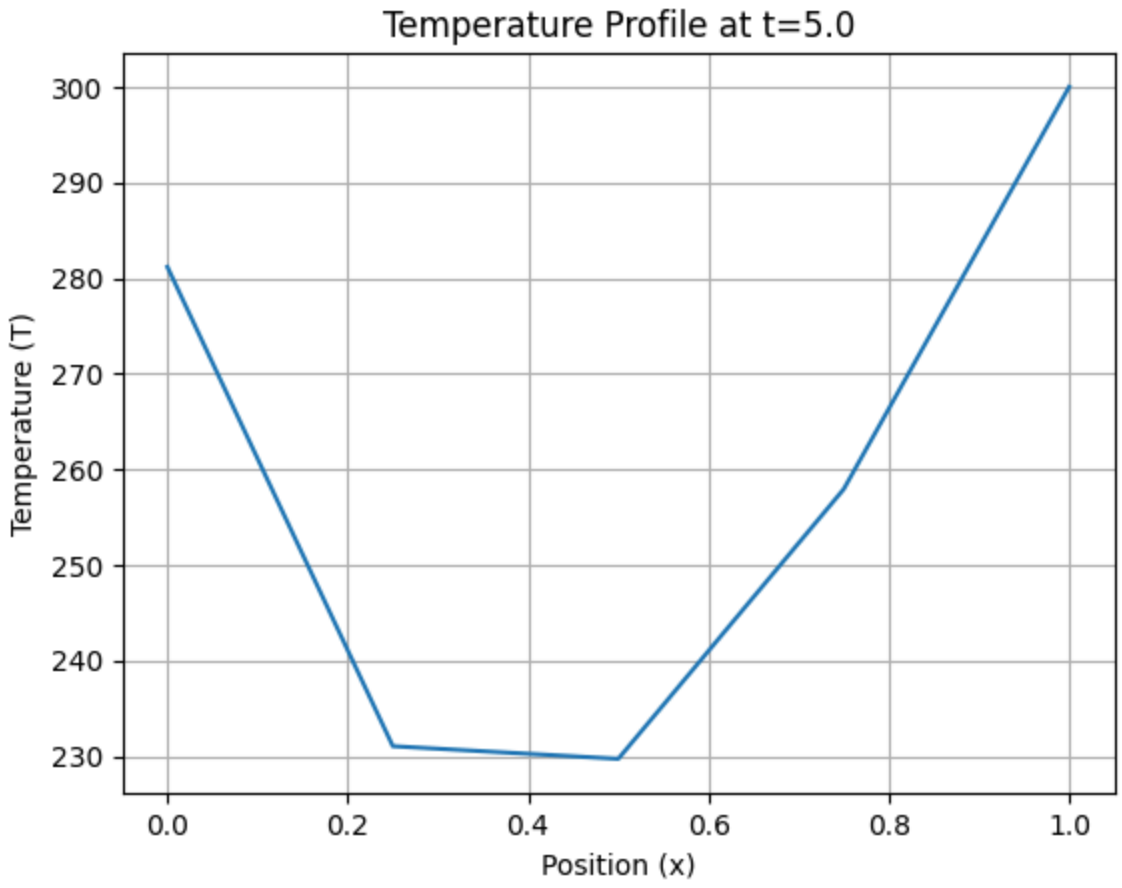
I. Frequency = 1Hz

Temperature at x=0.25L, t=5.0 = 229.71429824701715

Temperature at x=0.50L, t=5.0 = 257.9283187347199

Temperature at x=0.75L, t=5.0 = 300.0

Temperature at x=1.00L, t=5.0 = 300.0



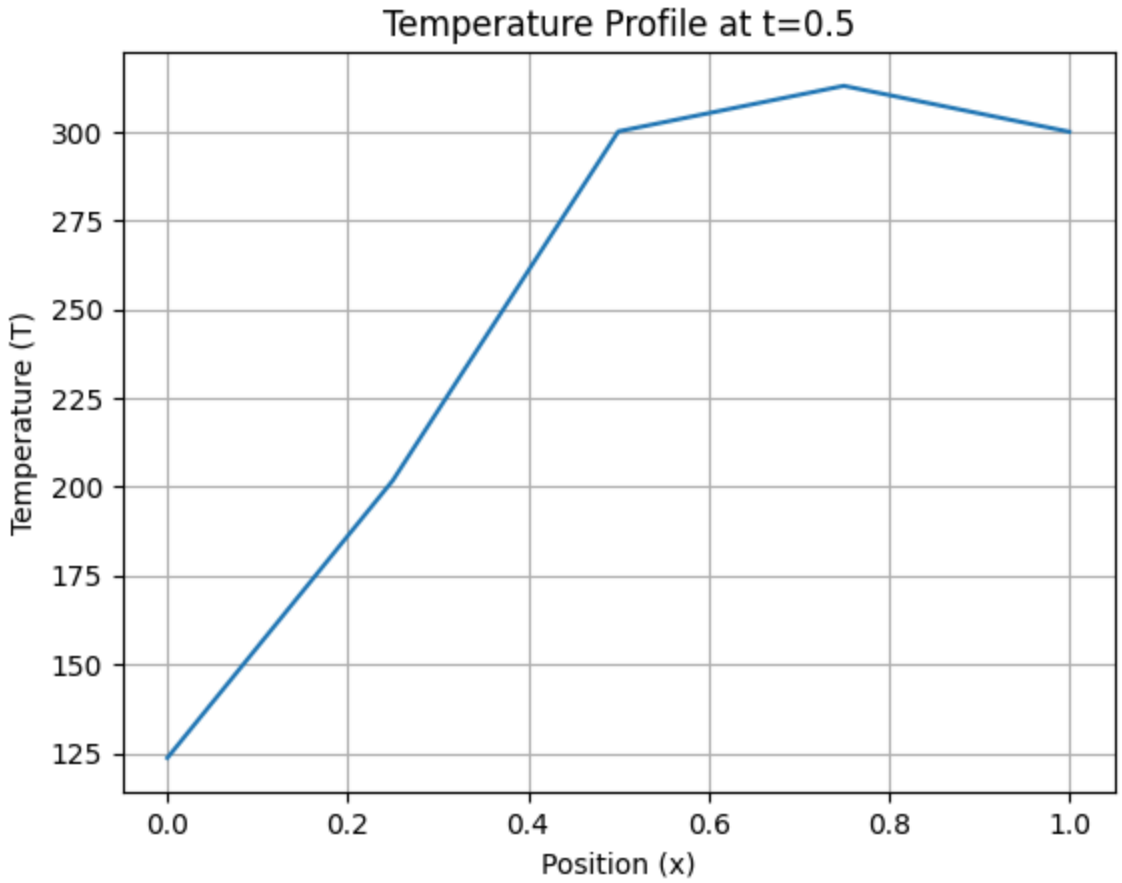
II. Frequency = 10Hz

Temperature at x=0.25L, t=0.5 = 300.0927310155166

Temperature at x=0.50L, t=0.5 = 312.9302969960695

Temperature at x=0.75L, t=0.5 = 300.0

Temperature at x=1.00L, t=0.5 = 300.0



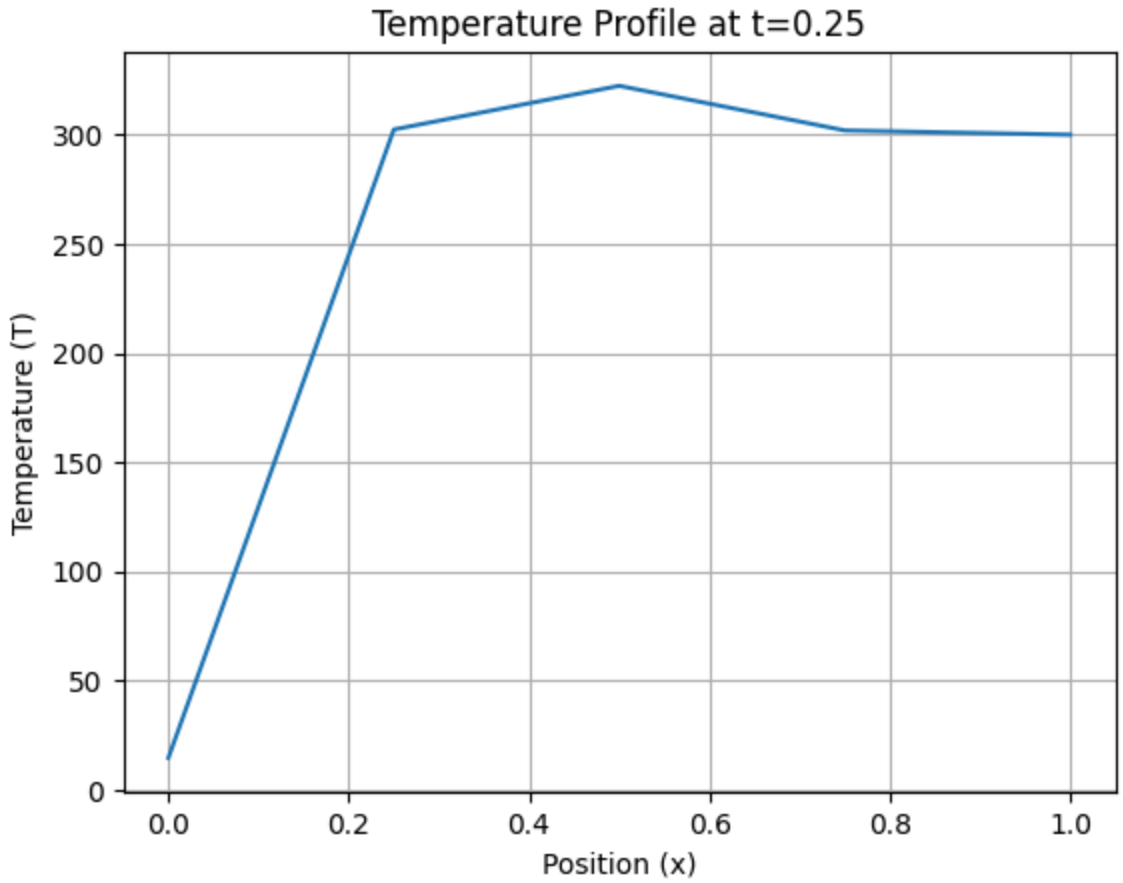
III. Frequency =20Hz

Temperature at x=0.25L, t=0.25 = 322.37538826839034

Temperature at x=0.50L, t=0.25 = 301.9618544511956

Temperature at x=0.75L, t=0.25 = 300.0

Temperature at x=1.00L, t=0.25 = 300.0



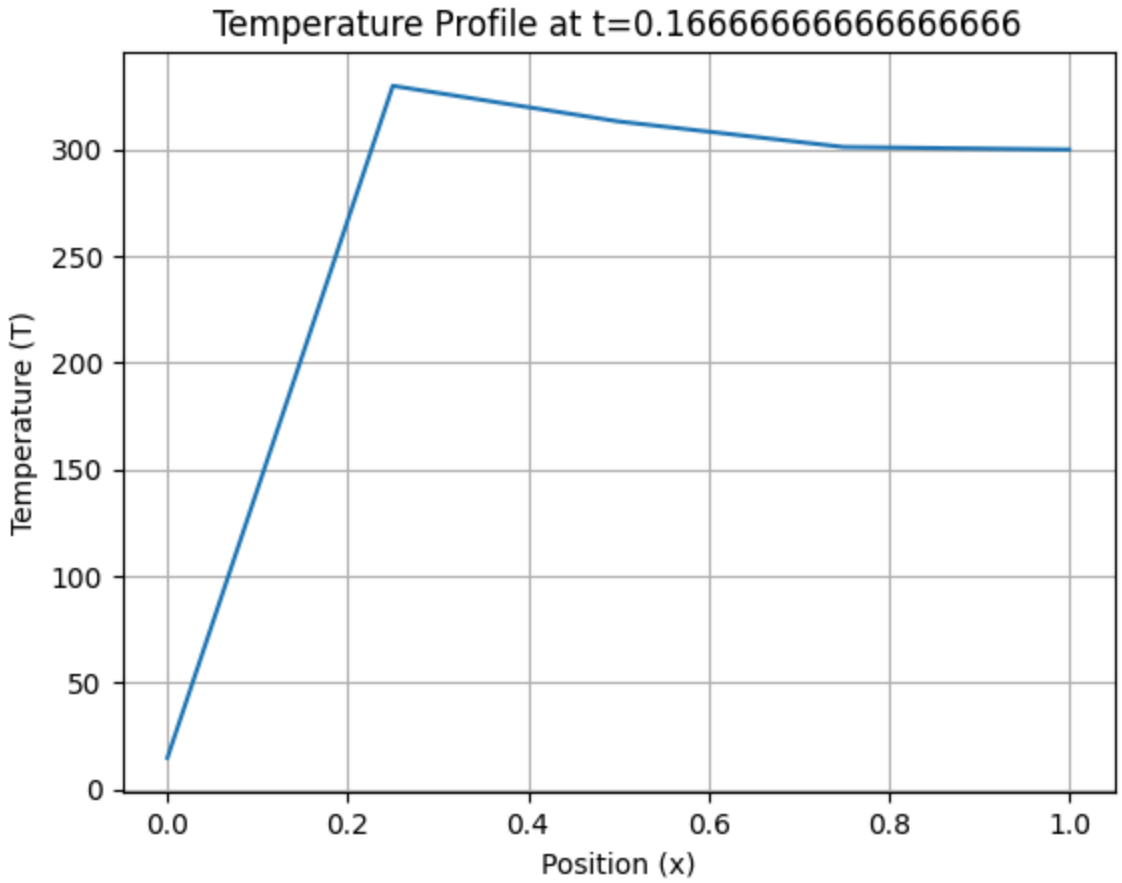
IV. Frequency = 30Hz

Temperature at x=0.25L, t=0.16666666666666666 = 313.21814641715673

Temperature at x=0.50L, t=0.16666666666666666 = 301.1887067599807

Temperature at x=0.75L, t=0.16666666666666666 = 300.0

Temperature at x=1.00L, t=0.16666666666666666 = 300.0



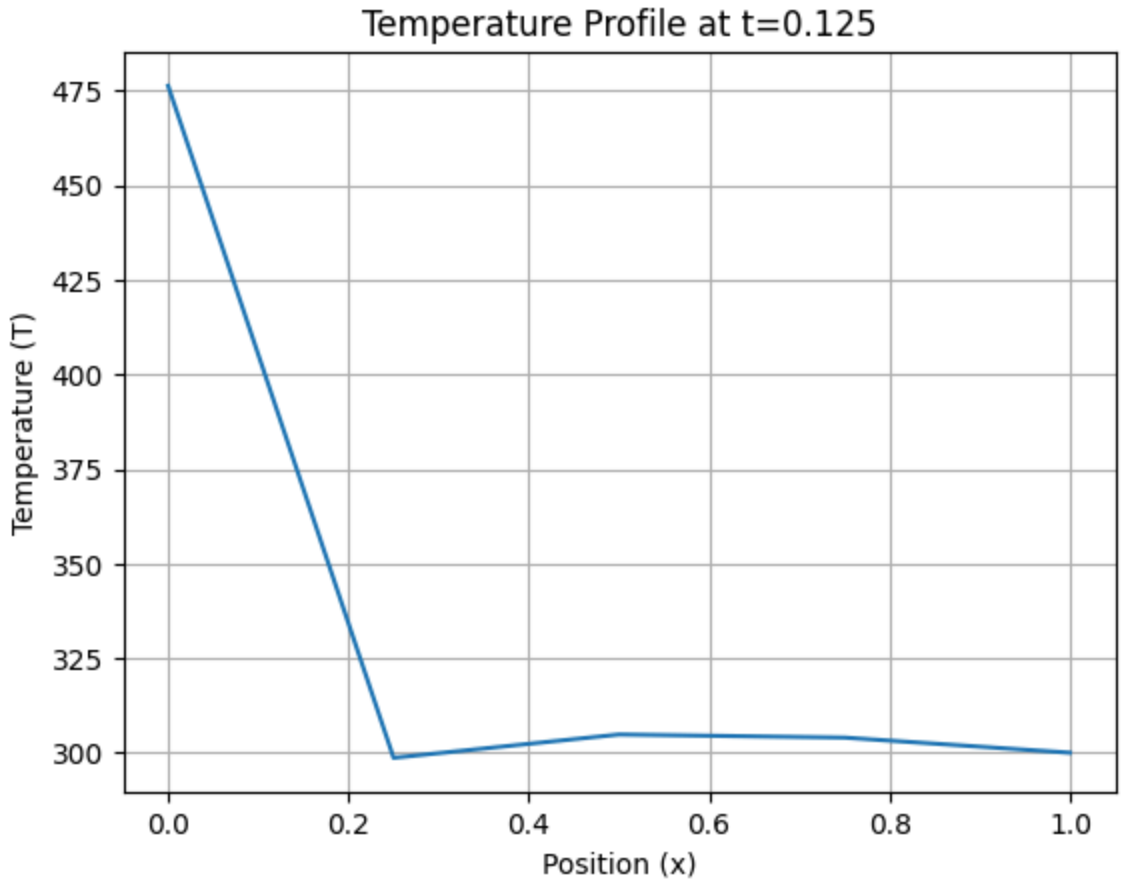
V. Frequency = 40Hz

Temperature at x=0.25L, t=0.125 = 304.81690968790053

Temperature at x=0.50L, t=0.125 = 303.99722231589146

Temperature at x=0.75L, t=0.125 = 300.0

Temperature at x=1.00L, t=0.125 = 300.0



Implicit Method

Discretized equation to solve for:

$$\rho Cp[T_i^{n=1} - T_i^n]/\Delta = K(T)[T^{n+1}_{i+1} - 2*T_i^{n+1} + T^{n+1}_{i-1}]$$

I. MATLAB Code:

For some reason I'm not able to get an exact graph via. Python code so, I shifted to Matlab code.

```
% Parameters
Cp = 235;      % Specific heat capacity
rho = 10500;   % Density
L = 1;         % Length of the domain (1 cm)
T0 = 300.0;    % Initial temperature
Tni = 300.0;   % Neumann boundary condition
dx = 0.25;     % Spatial step size

% Specify frequency of the sinusoidal boundary condition
frequency = 1;  % 1 Hz as an example

% Calculate final time for at least 5 cycles
num_cycles = 5;
final_time = num_cycles / frequency;

% Temporal parameters
dt = 0.01;  % Temporal step size
num_steps = round(final_time / dt);  % Number of time steps

% Initialize temperature array
nx = round(L / dx) + 1;
T = ones(nx - 1, num_steps) * T0;  % Note: U_n is of dimension (nx-1) x num_steps

% Initialize stability flag
stable = true;

% Function to calculate k(T) at each spatial and time step
k = @(T) -0.0024*T.^2 + 0.99*T + 287.1191;

% Initialize coefficient matrix A using sparse matrix
s = dt / (rho * Cp * dx^2);
```

```
A = speye(nx - 1) - s * gallery('tridiag', nx - 1, -1, 2, -1);

% Iterate through time steps
for j = 2:num_steps
    % Update boundary conditions with a sinusoidal profile
    omega = 2 * pi * frequency;
    T(1, j) = T0 * (1 + sin(omega * j * dt));
    T(end, j) = Tni;

    % Evaluate thermal conductivity function k(T) at each spatial and time step
    k_values = k(T(:, j-1));

    % Construct right-hand-side vector b
    b = T(:, j-1) + s * k_values;

    % Solve the system of equations using the backslash operator
    T(:, j) = A \ b;
end

% Display results
disp(['Temperature at x=0.25L, t=' num2str(final_time) ' = ' num2str(T(round(0.25*L/dx),
end))]);
disp(['Temperature at x=0.5L, t=' num2str(final_time) ' = ' num2str(T(round(0.5*L/dx),
end))]);
disp(['Temperature at x=0.75L, t=' num2str(final_time) ' = ' num2str(T(round(0.75*L/dx),
end))]);
disp(['Temperature at x=L, t=' num2str(final_time) ' = ' num2str(T(end, end))]);

% Plot the final temperature profile
x_values = linspace(0, L, nx);
plot(x_values, [T; Tni * ones(1, num_steps)]);
title(['Temperature Profile at t=' num2str(final_time)]);
xlabel('Position (x)');
ylabel('Temperature (T)');
grid on;
```

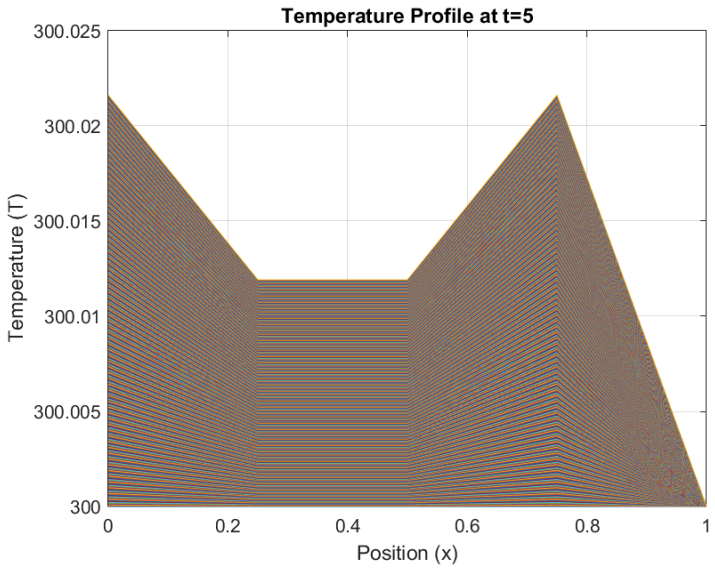
II. Frequency = 1Hz:

Temperature at x=0.25L, t=5 = 300.0216

Temperature at x=0.5L, t=5 = 300.0119

Temperature at x=0.75L, t=5 = 300.0119

Temperature at x=L, t=5 = 300.0216



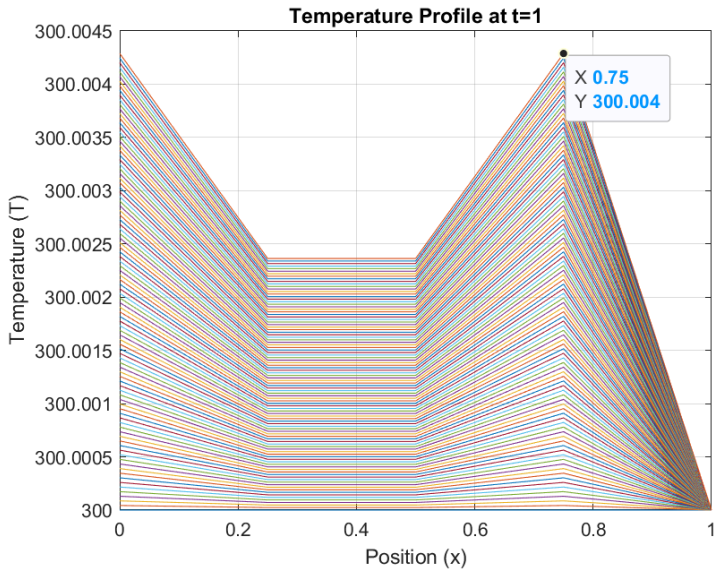
III. Frequency = 5Hz:

Temperature at x=0.25L, t=1 = 300.0043

Temperature at x=0.5L, t=1 = 300.0024

Temperature at x=0.75L, t=1 = 300.0024

Temperature at x=L, t=1 = 300.0043



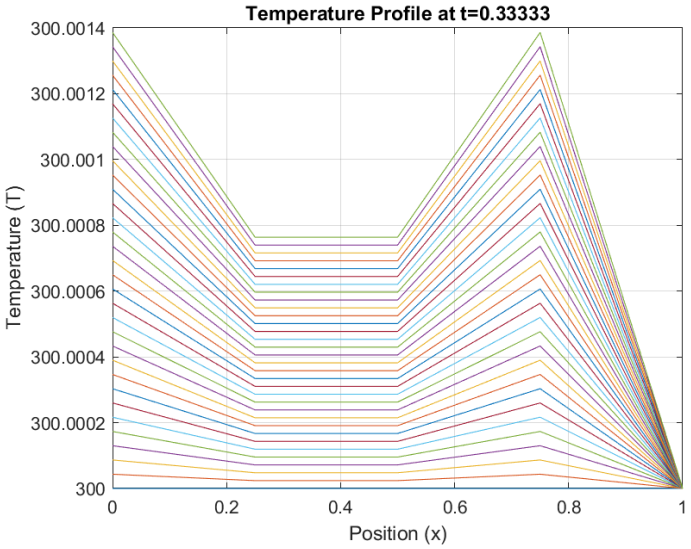
IV. Frequency = 15Hz:

Temperature at x=0.25L, t=0.33333 = 300.0014

Temperature at x=0.5L, t=0.33333 = 300.0008

Temperature at x=0.75L, t=0.33333 = 300.0008

Temperature at x=L, t=0.33333 = 300.0014



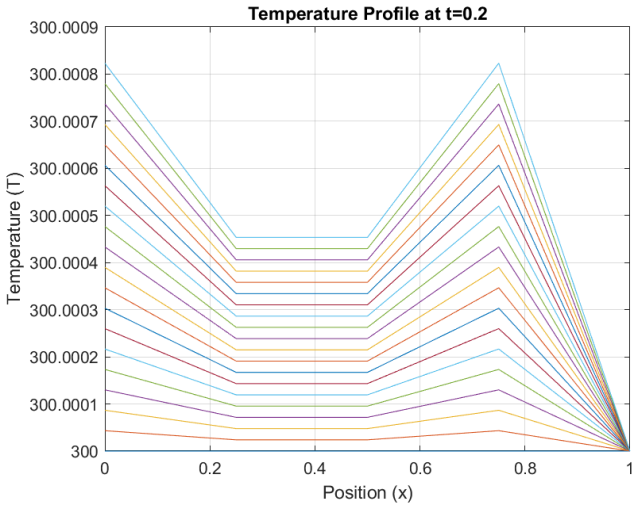
V. Frequency = 25Hz:

Temperature at x=0.25L, t=0.2 = 300.0008

Temperature at x=0.5L, t=0.2 = 300.0005

Temperature at x=0.75L, t=0.2 = 300.0005

Temperature at x=L, t=0.2 = 300.0008



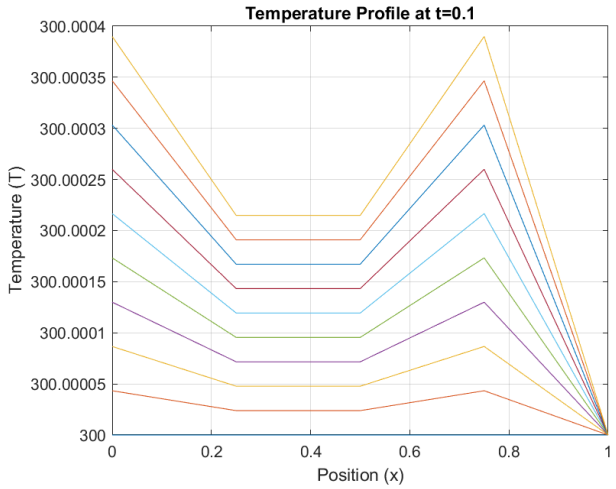
VI. Frequency = 50Hz:

Temperature at x=0.25L, t=0.1 = 300.0004

Temperature at x=0.5L, t=0.1 = 300.0002

Temperature at x=0.75L, t=0.1 = 300.0002

Temperature at x=L, t=0.1 = 300.0004



Crank-Nicolson Method

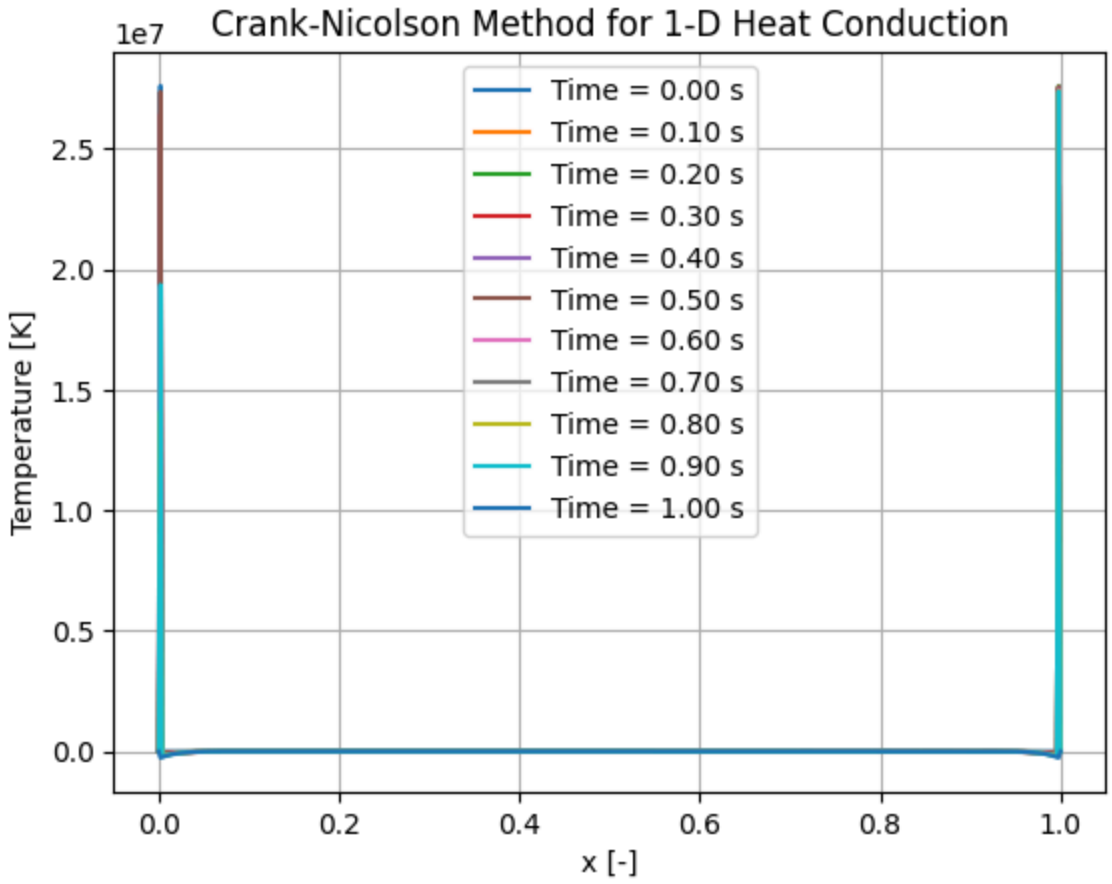
I. Frequency = 1Hz

Temperature at delta x = 0.25L: 314.73 K

Temperature at delta x = 0.5L: 300.09 K

Temperature at delta x = 0.75L: 314.88 K

Temperature at delta x = L: 300.00 K



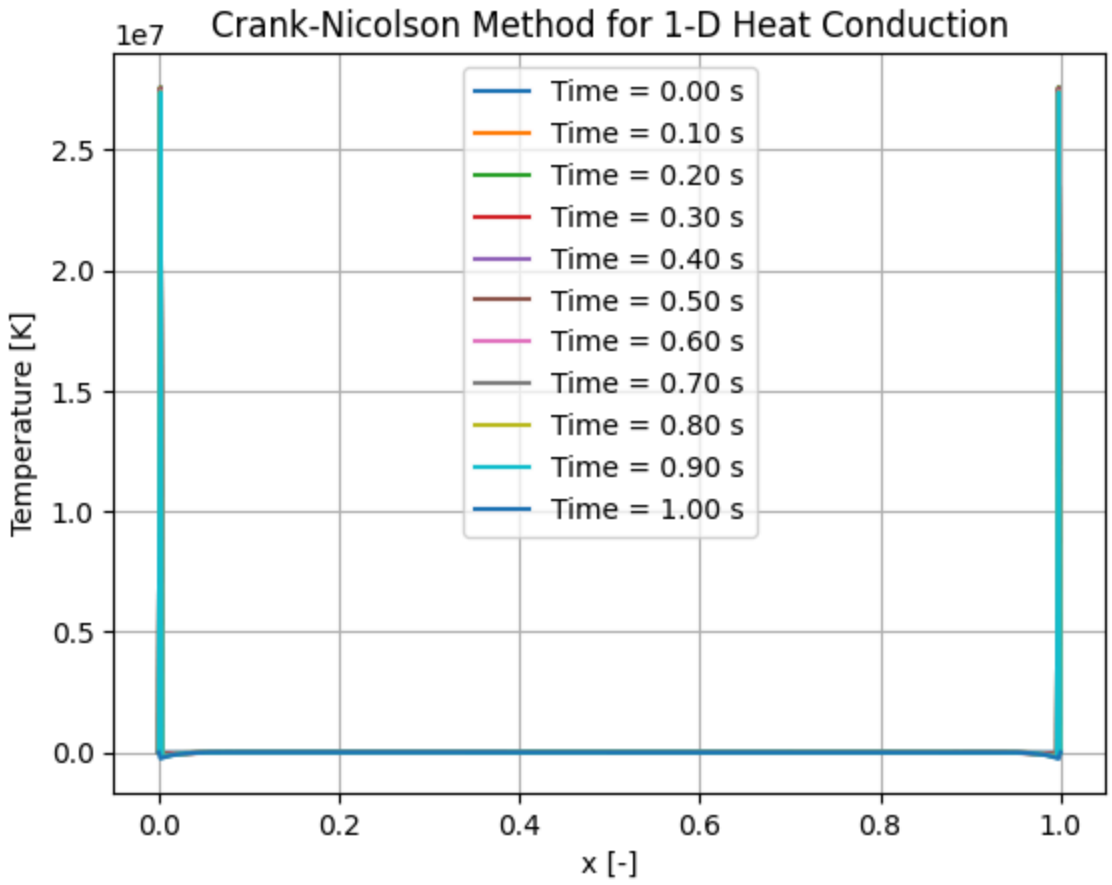
II. Frequency = 10Hz

Temperature at delta x = 0.25L: 312.26 K

Temperature at delta x = 0.5L: 297.02 K

Temperature at delta x = 0.75L: 312.80 K

Temperature at delta x = L: 300.00 K



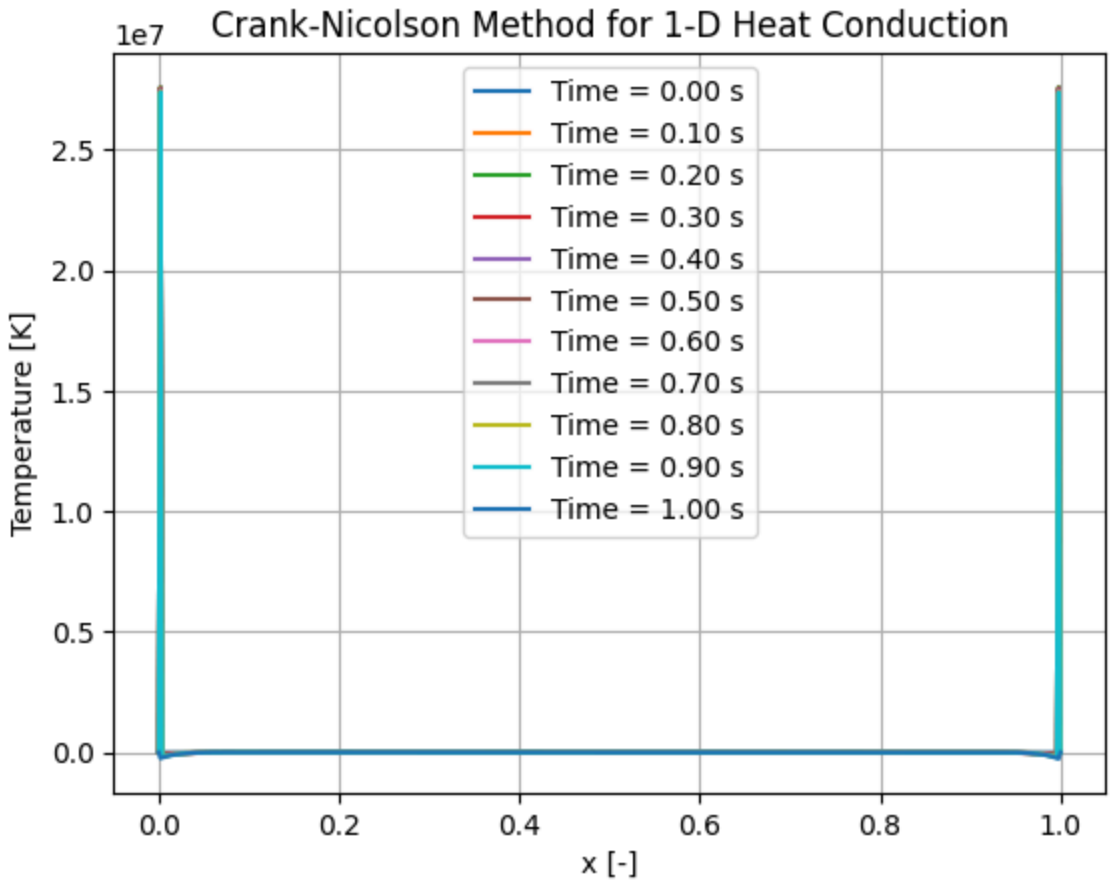
III. Frequency = 20Hz

Temperature at delta x = 0.25L: 311.59 K

Temperature at delta x = 0.5L: 294.26 K

Temperature at delta x = 0.75L: 310.58 K

Temperature at delta x = L: 300.00 K



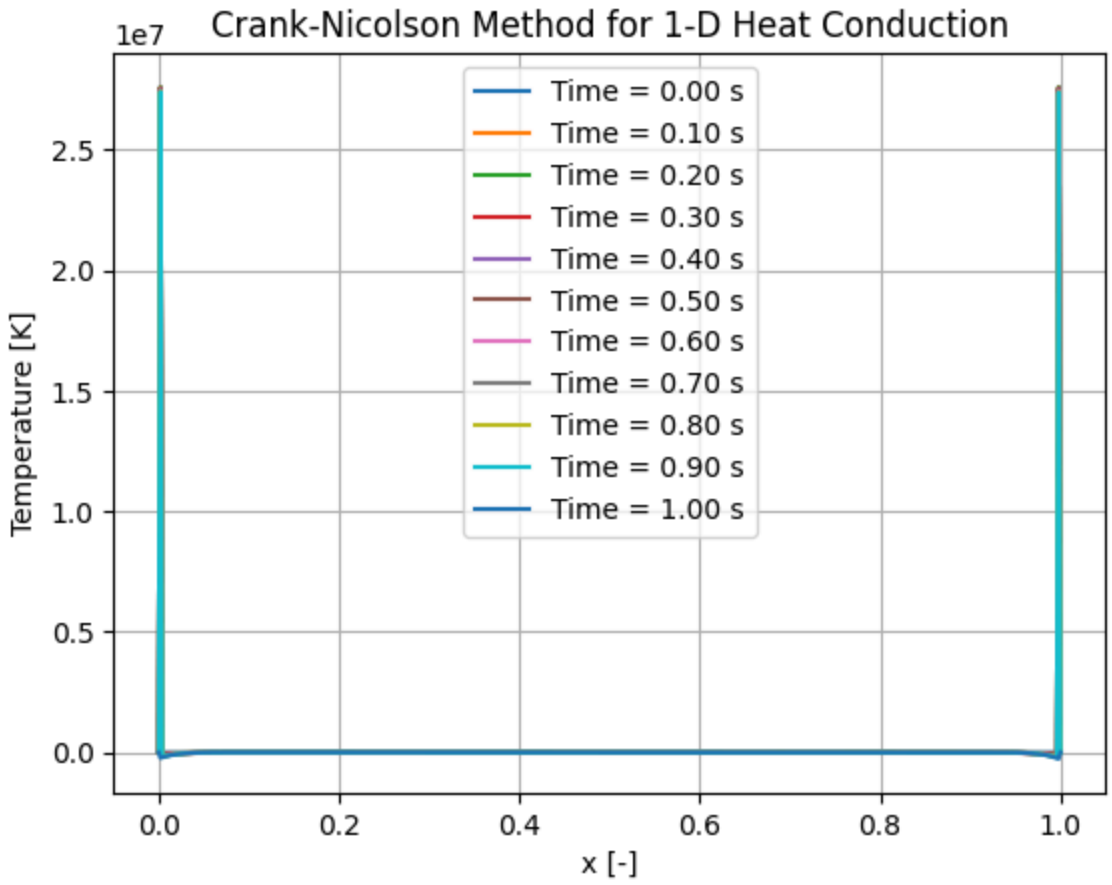
IV. Frequency = 30Hz

Temperature at delta x = 0.25L: 312.51 K

Temperature at delta x = 0.5L: 291.26 K

Temperature at delta x = 0.75L: 307.83 K

Temperature at delta x = L: 300.00 K



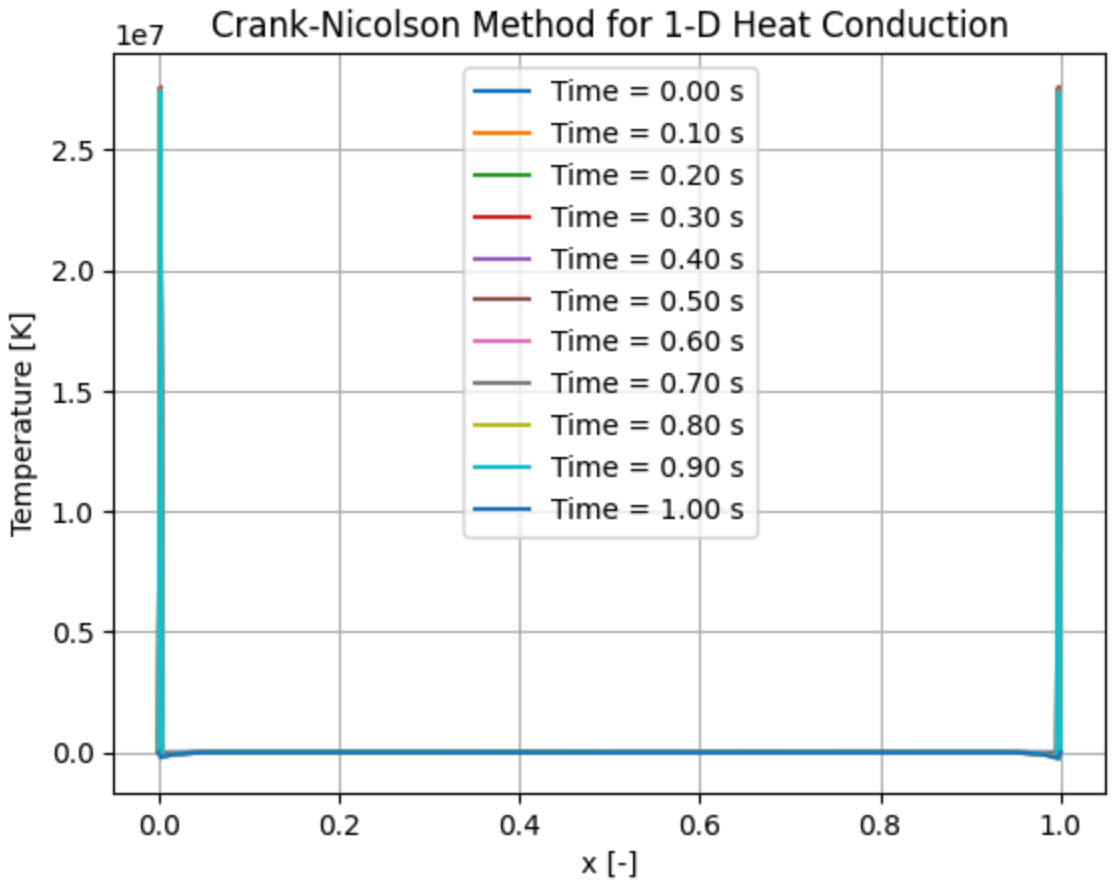
V. Frequency = 40Hz

Temperature at delta x = 0.25L: 313.72 K

Temperature at delta x = 0.5L: 286.21 K

Temperature at delta x = 0.75L: 303.43 K

Temperature at delta x = L: 300.00 K



Gauss Elimination Method:

Output:

Spatial Step(dx)	Number of internal grid points	Computational Time
0.1	10	0.543173
0.01	100	25.913634

I refrained from conducting computations for dx values such as 0.001 or 0.00001 due to the substantial computational slowdown. The execution time became excessively prolonged, rendering the process unfeasible to yield conclusive results.
It's noteworthy to mention that the use of Python, being a relatively slower language compared to some others, significantly contributed to the extended computational duration. For instance, even for a moderately small value like 0.01, the computation demanded 25 seconds to complete.

Gauss Jordan Method:

Output:

Spatial Step(dx)	Number of internal grid points	Computational Time
0.1	10	0.658703
0.01	100	28.043992

Excluding computations for dx values such as 0.001 or 0.00001 was a deliberate decision, as the computational process exhibited significant sluggishness, rendering it impractical to achieve a conclusive outcome. This is especially pronounced due to Python's inherent slower processing speed, as evidenced by the extended time required even for a relatively moderate value like 0.01, which consumed 25 seconds for computation.

Thomas Algorithm:

Output:

Spatial Step(dx)	Number of internal grid points	Computational Time
0.1	10	0.279860
0.01	100	0.378863
0.001	1000	1.386240

Result:

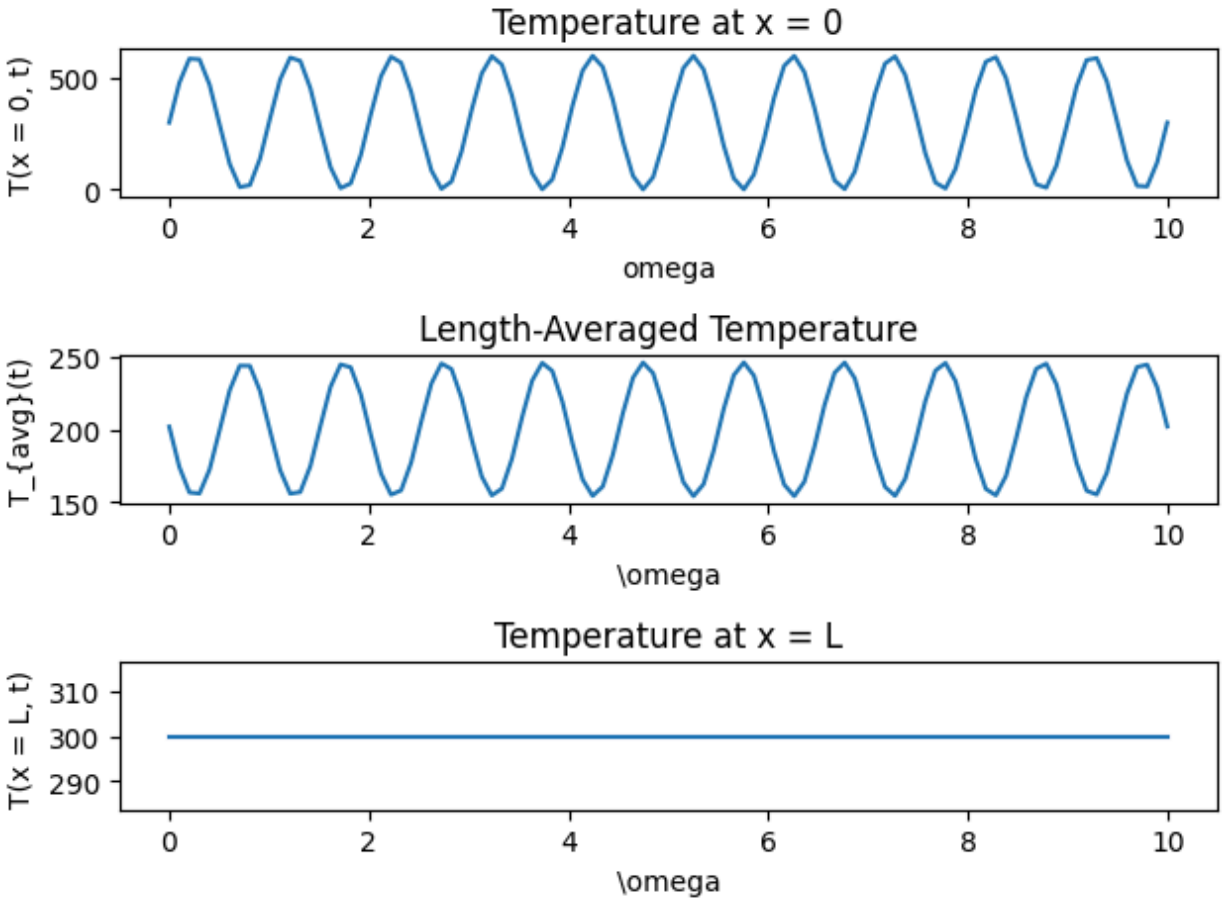
The results derived from the provided table lead to the conclusion that among the three algorithms, Thomas Algorithm demonstrates superior speed, evident from the computational time. Nevertheless, it is noteworthy that all three algorithms faced challenges when tasked with computing values for a significantly large number of internal grid points.

Part (f):

I. Output1:

Spatial Step (dx) = 0.1

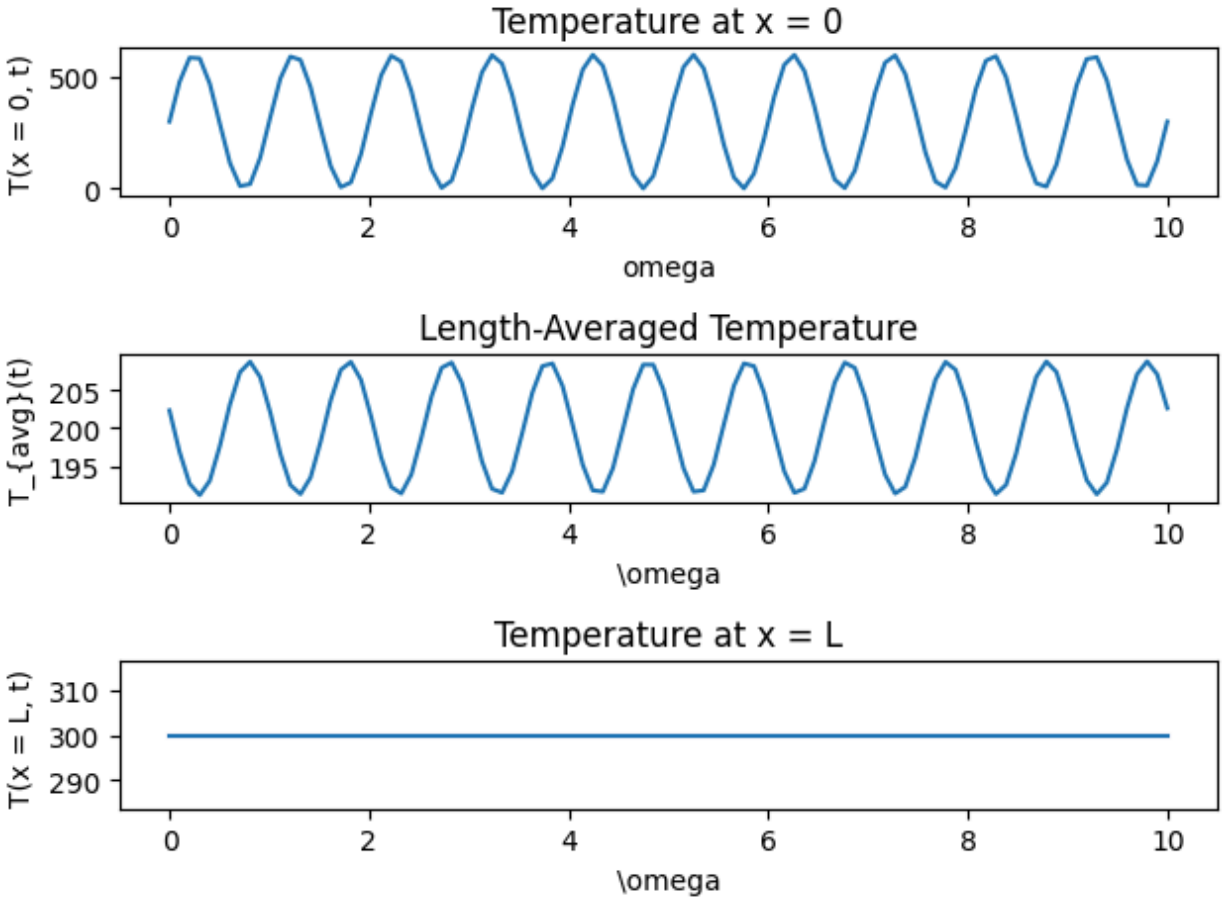
Number of Internal Grid Points: 10



II. Output2:

Spatial Step (dx) = 0.01

Number of Internal Grid Points: 100



III. Output3:

Spatial Step (dx) = 0.001

Number of Internal Grid Points: 1000