ME3180

FEM & CFD Theory

Assignment 1

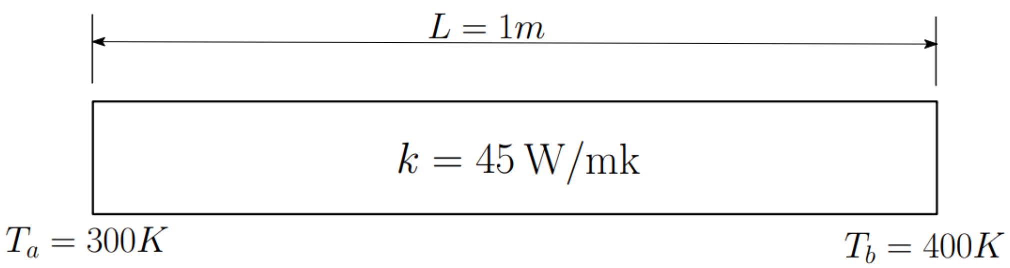
Abhishek Ghosh

ME21BTECH11001

1D Steady State Heat Conduction Equation

Part 1

Consider a metal rod of length 1 meter with thermal conductivity (k = 45 W/m.K) shown in Fig 1. Both of its ends are maintained at a constant temperature of Ta = 300 K and Tb = 400 K, respectively. Assuming that the heat transfer is only taking place along the length of the rod, solve the steady state 1D heat conduction equation numerically and plot the steady state contour of the temperature distribution along the length of the rod. Use a second-order central difference scheme to discretize your governing partial differential equation.

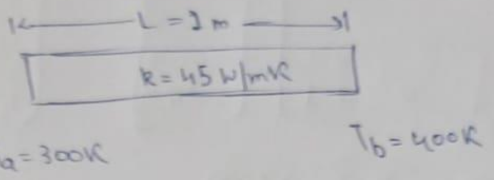


Theory:

ME3180 Assignment 1

Abhishek Ghosh
ME21BTECH11001

Question 1



Assuming 1D heat transfer

Governing eqn : $\frac{\partial^2 T}{\partial x^2} = 0$ $T(0) = 300K$
$T(L) = 400K$

Analytical

$$\int \frac{\partial^2 T}{\partial x^2} = 0$$

$$\Rightarrow \frac{\partial T}{\partial x} = c_1$$

$$\Rightarrow T(x) = c_1 x + c_2$$

@ $x=0$ $T=300 \Rightarrow c_2 = 300$
@ $x=L$ $T=400 \Rightarrow c_1 = 100$

$$\Rightarrow T(x) = 100x + 300$$

Discretization via central diff method :—

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1} - 2T_i + T_{i+1}}{h^2} = 0 \Rightarrow T_i = \frac{1}{2}(T_{i-1} + T_{i+1})$$

$$\left(\frac{1}{h^2}\right)T_{i-1} + \left(\frac{-2}{h^2}\right)T_i + \left(\frac{1}{h^2}\right)T_{i+1} = 0$$

$$\begin{bmatrix} 1 & 0 & 0 & & & \\ \cancel{1/h^2} & \cancel{-2/h^2} & \cancel{1/h^2} & \cdots & & \\ & 1/h^2 & -2/h^2 & 1/h^2 & \cdots & \\ & & \cdots & & & \\ & & & & & \\ & & & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ \vdots \\ \vdots \\ T_n \end{Bmatrix} = \begin{Bmatrix} 300 \\ 0 \\ \vdots \\ \vdots \\ 400 \end{Bmatrix}$$

with $T_1 = 300$ & $T_n = 400$

Jacobi $\longrightarrow$ for k^{th} iteratn

$$T_i^k = \frac{1}{2}\left[T_{i-1}^{k-1} + T_{i+1}^{k-1}\right]$$

Gauss Siedel $\longrightarrow$

$$T_i^k = \frac{1}{2}\left[T_{i-1}^{k} + T_{i+1}^{k-1}\right]$$

TDMA $\longrightarrow$

$$2T_i = T_{i+1} + T_{i-1}$$

$a = 2 \quad b = 1 \quad c = 1 \quad d = 0 \quad [a_i T_i = b_i T_{i+1} + c_i T_{i-1} + d_i]$

for $2 \longrightarrow N$

$$P_i = \frac{b_i}{a_i - c_i P_{i-1}} \qquad Q_i = \frac{d_i + c_i Q_{i-1}}{a_i - c_i P_{i-1}}$$

end

with $P_1 = b_1/a_1$ & $Q_1 = d_1/a_1$

$$Q_N = Q_N$$

for $N-1 \longrightarrow 1$

$$Q_i = P_i Q_{i+1} + Q_i$$

end

where $Q(N) = 400K$ & $Q(0) = 300K$.

For non-uniform grid :-

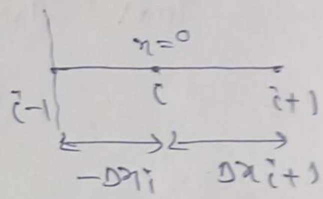
i=1 ———— i=N

x=0 ———— x=L

n > 1 finer mesh near x=0

n < 1 finer mesh near x=L

$$\frac{x_i}{L} = \left(\frac{i-1}{N-1}\right)^n$$



$$R = \frac{\Delta x_{i+1}}{\Delta x_i}$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1} + R T_{i-1} - (1+R) T_i}{\Delta x^2 (R+1) R} = 0$$

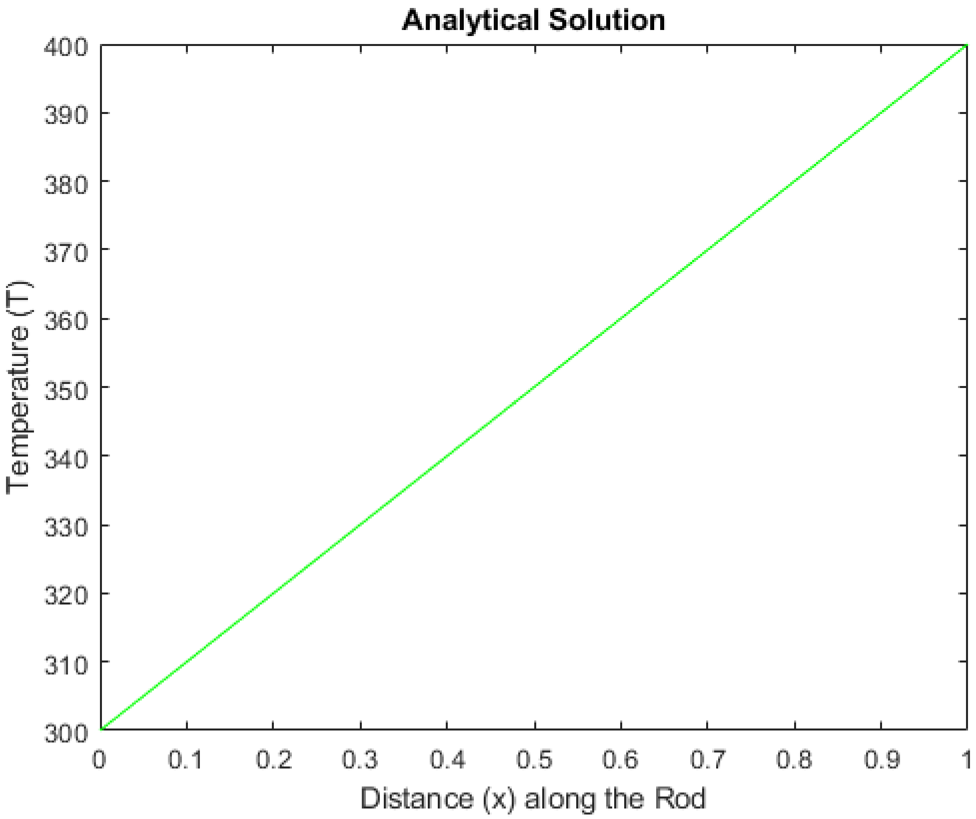
$$\Rightarrow \quad T_i = \frac{T_{i+1} + R T_{i-1}}{1+R}$$

Code & Plots:

Initial conditions

```
% Abhishek Ghosh
% ME21BTECH11001

clc
clear all
% Given parameters
L = 1;
k = 45;
Ta = 300;
Tb = 400;
n = 25;

% 1D problem
initial_position = 0;
final_position = L;

h = (final_position - initial_position)/(n-1);
x = linspace(0, L, n);

% Analytical Solution
T_analytical = 100 * x + 300;
x_analytical = x;

plot(x_analytical, T_analytical, 'g-');
title('Analytical Solution');
```

```
xlabel('Distance (x) along the Rod');
ylabel('Temperature (T)');
```

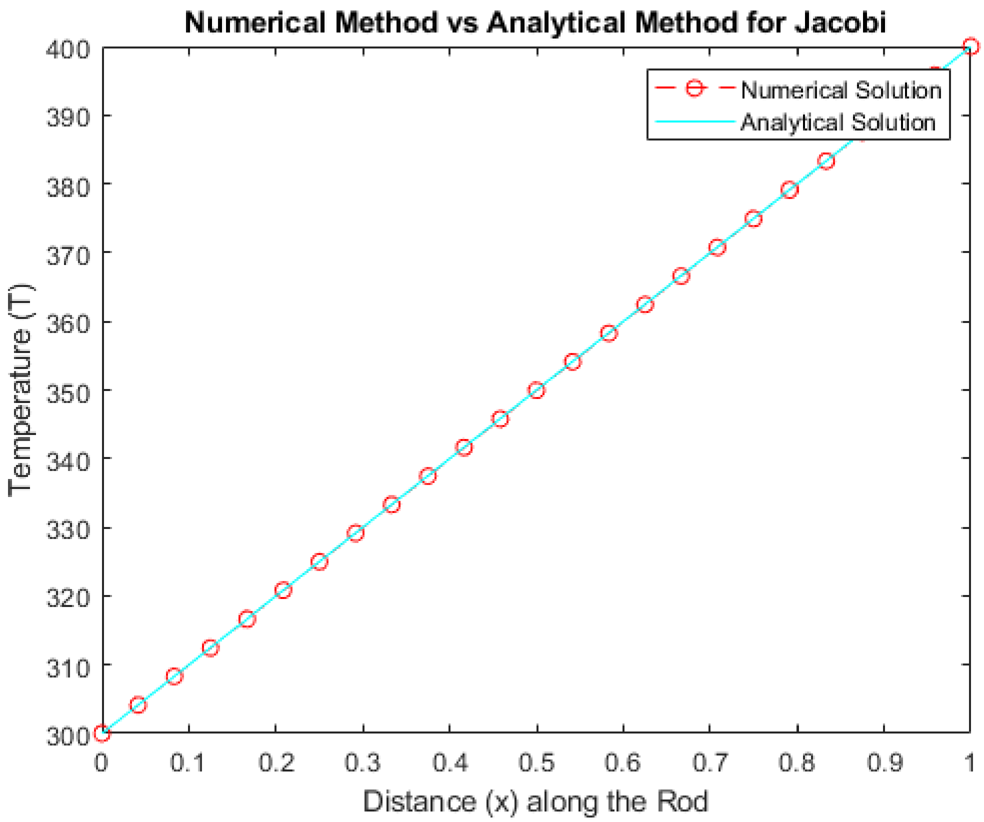


Jacobi Method

```
% Jaacobi Method

T = zeros(1, n);
% Initializing Boundary Conditions
T(1) = Ta;
T(n) = Tb;

% Max threshold for acceptable error
Tolerance = 1e-3;
T_old = T;

% Variable to keep track of the number of iterations performed by the algorithm
iterations = 0;
Error = 1;

while Error > Tolerance
    for i = 2:n-1 % Central Difference Scheme
        T(i) = 0.5 * (T_old(i-1) + T_old(i+1));
    end

    Error = max(abs(T - T_old));
    T_old = T;
    iterations = iterations + 1;
end


% Comparing Jacobi wiht analytical Solution
figure;
plot(x, T, 'r--o');
```

```
hold on;
plot(x_analytical, T_analytical, 'c-');

title('Numerical Method vs Analytical Method for Jacobi');
xlabel('Distance (x) along the Rod');
ylabel('Temperature (T)');
legend('Numerical Solution', 'Analytical Solution');


disp(['No. of Iterations in Jacobi Method: ', num2str(iterations)]);
```

```
No. of Iterations in Jacobi Method: 1041
```

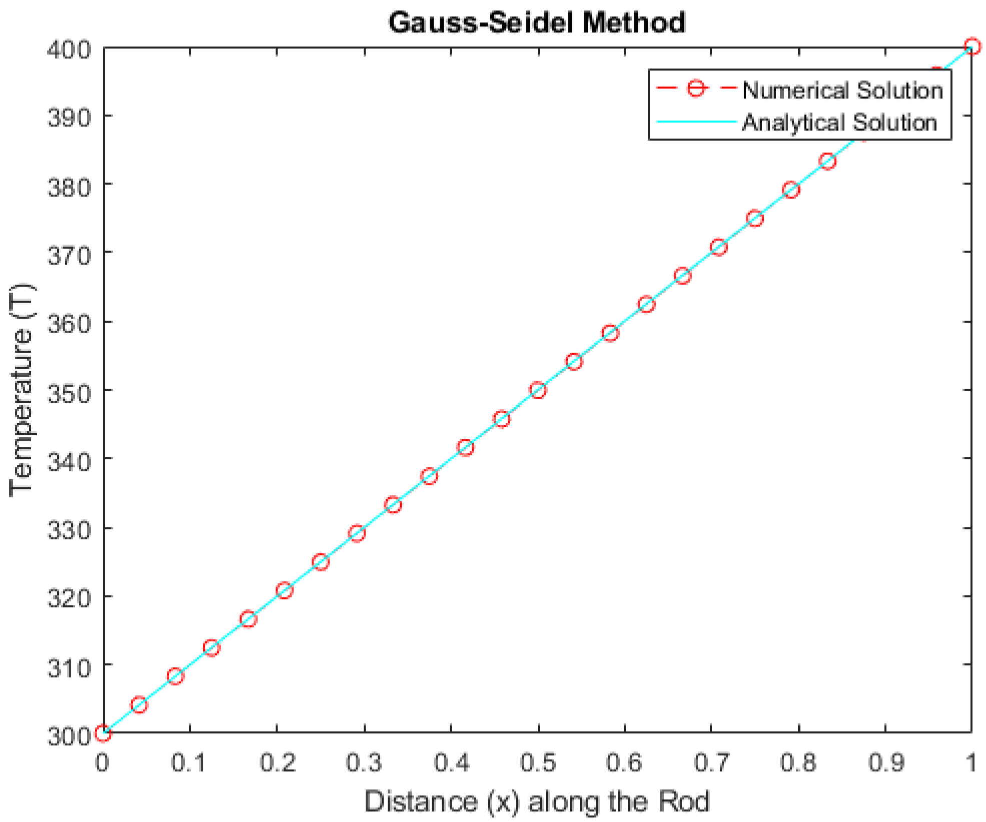


Gauss Siedel Method

```
% Gauss-Seidel Method
T_gs = zeros(1, n);
T_gs(1) = Ta;
T_gs(n) = Tb;

T_old_gs = T_gs;
iterations = 0;
Error = 1;

% Arrays to store errors and iterations
Errors_gs = [];
iterate_gs = [];

while Error > Tolerance
```

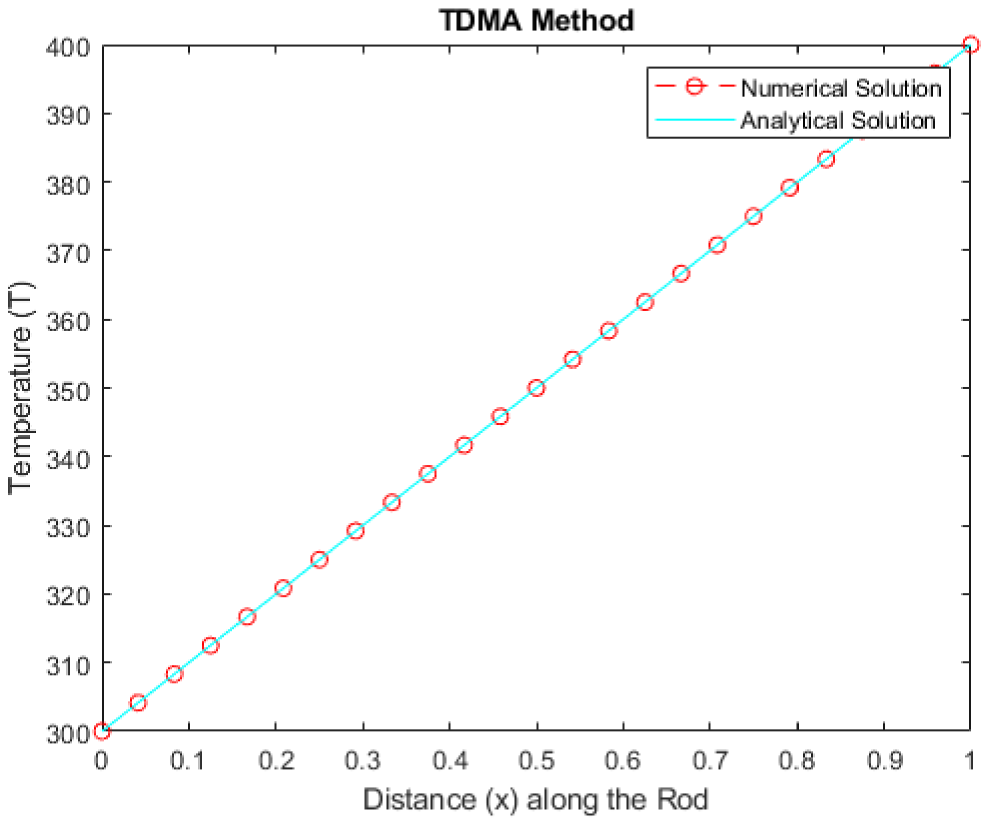
```
    for i = 2:n-1
        T_gs(i) = 0.5 * (T_gs(i-1) + T_old_gs(i+1));
    end

    Error = max(abs(T_gs - T_old_gs));

    iterations = iterations + 1;
    iterate_gs = [iterate_gs, iterations];

    Errors_gs = [Errors_gs, (norm(T_gs - T_old_gs, 2) / norm(T_old_gs, 2))];

    T_old_gs = T_gs;
end

% Plotting the results
x = linspace(0, 1, n);
figure;
plot(x, T_gs, 'r--o');
hold on;
plot(x_analytical, T_analytical, 'c-');

title("Gauss-Seidel Method");
ylabel("Temperature (T)");
xlabel("Distance (x) along the Rod");
legend('Numerical Solution', 'Analytical Solution');

disp(['No. of Iterations in Gauss-Seidel Method: ', num2str(iterations)]);
```

```
No. of Iterations in Gauss-Seidel Method: 523
```



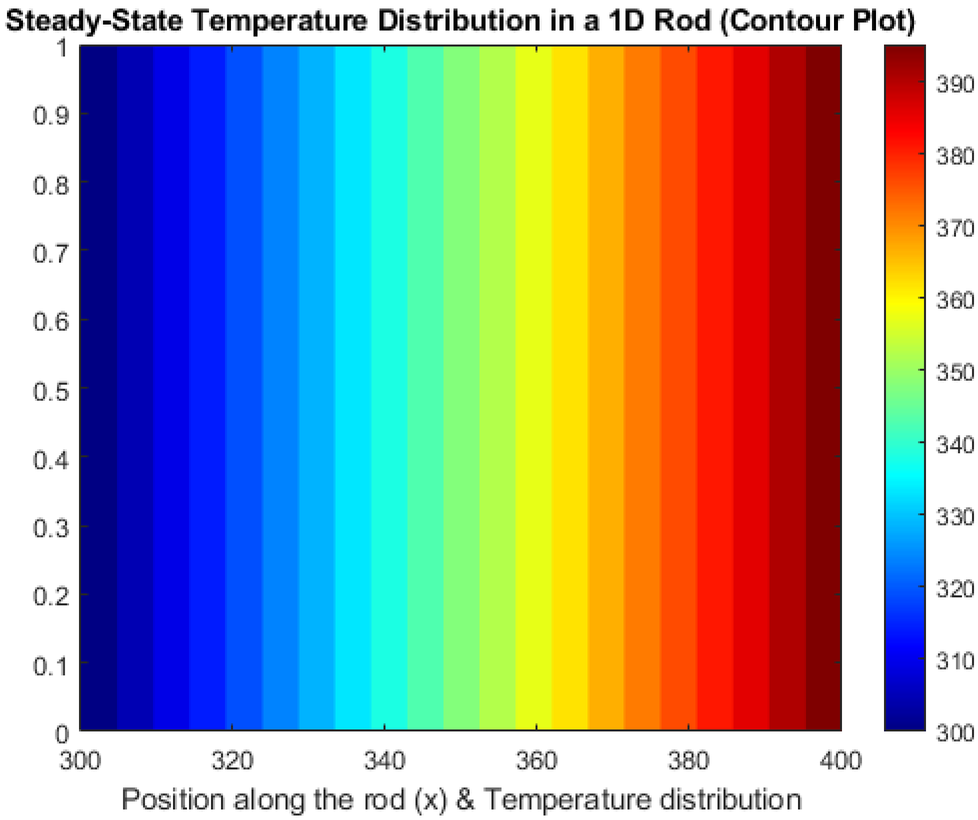
TDMA

```
% TDMA
T = zeros(1, n);
T(1) = Ta;
T(n) = Tb;

P = zeros(1, n);
Q = zeros(1, n);

a = 2;
b = 1;
c = 1;
d = 0;

P(1) = 0;
Q(1) = Ta;

for i = 2:n-1
    P(i) = b / (a - c * P(i - 1));
    Q(i) = (d + c * Q(i - 1)) / (a - c * P(i - 1));
end

Q(n) = Tb;

for i = n-1:-1:1
    T(i) = T(i + 1) * P(i) + Q(i);
end
figure;
plot(x, T, 'r--o');
hold on;
plot(x_analytical, T_analytical, 'c-');

title('TDMA Method');
xlabel('Distance (x) along the Rod');
ylabel('Temperature (T)');
legend('Numerical Solution', 'Analytical Solution');

disp(['No. of Iterations in TDMA Method: ', num2str(iterations)]);
```



Contour Plot (Temperature distribution along the rod)

```
% PLotting of Heat Distribution along the rod

% Create meshgrid
[X, Y] = meshgrid(x, T_gs);

% Create the contour plot
figure;
contourf(Y, X, Y, 20, 'LineColor', 'none');
colormap(jet);
colorbar();
xlabel('Position along the rod (x) & Temperature distribution');
% ylabel('Temperature (T)');
title('Steady-State Temperature Distribution in a 1D Rod (Contour Plot)');
```



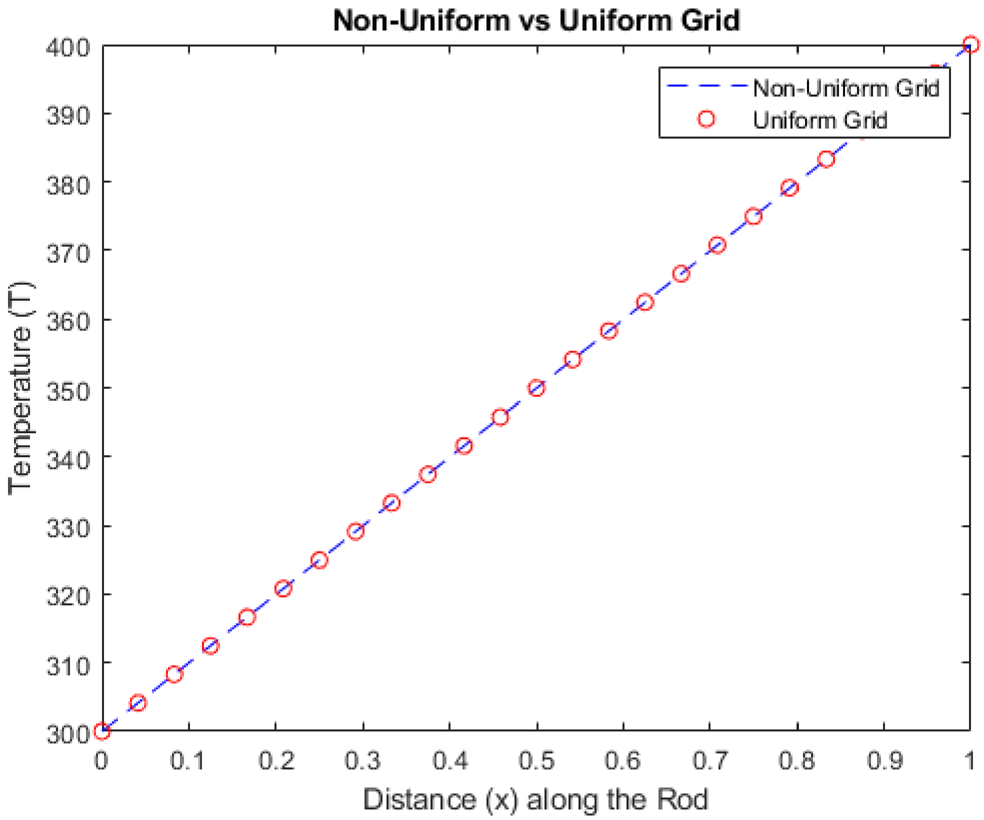
As we keep on going to the left the temperature keeps on increasing linearly.

Uniform vs non-uniform grid size

```
% Uniform vs Non Uniform grid
% Non-uniform grid generation
p = 0.3;
x_nu = zeros(1, n);
for i = 2:n
    x_nu(i) = L * ((i - 1) / (n - 1))^p;
end

% Initializing the temperature array for non-uniform grid
T_nu = zeros(1, n);
T_nu(1) = Ta;
T_nu(n) = Tb;

T_old_nu = T_nu;

iterations = 0;
Error = 1;
Errors_nu = [];
iterate_nu = [];

while Error > Tolerance
    for i = 2:n-1
        R = (x_nu(i+1) - x_nu(i)) / (x_nu(i) - x_nu(i-1));
        T_nu(i) = (R * T_nu(i-1) + T_old_nu(i+1)) / (1 + R);
    end
    Error = max(abs(T_nu - T_old_nu));
```

```
    iterations = iterations + 1;
    iterate_nu = [iterate_nu, iterations];

    % Storing error for non-uniform grid, L2 norm is used
    Errors_nu = [Errors_nu, (norm(T_nu - T_old_nu, 2) / norm(T_old_nu, 2))];

    T_old_nu = T_nu;
end

% Uniform grid using Gauss-Seidel
T_gs = zeros(1, n);
T_gs(1) = Ta;
T_gs(n) = Tb;

T_old_gs = T_gs;
iterations_gs = 0;
Error_gs = 1;

while Error_gs > Tolerance
    for i = 2:n-1
        T_gs(i) = 0.5 * (T_gs(i-1) + T_old_gs(i+1));
    end
    Error_gs = max(abs(T_gs - T_old_gs));

    iterations_gs = iterations_gs + 1;
    T_old_gs = T_gs;
end

% Plotting the results
x = linspace(0, L, n);
figure;
plot(x_nu, T_nu, 'b--');
hold on;
plot(x, T_gs, 'ro');
title("Non-Uniform vs Uniform Grid");
ylabel("Temperature (T)");
xlabel("Distance (x) along the Rod");
legend(["Non-Uniform Grid", "Uniform Grid"]);

disp(['No. of Iterations in Gauss Seidel Method with Non-Uniform Grid: ',
num2str(iterations)]);
disp(['No. of Iterations in Gauss Seidel Method with Uniform Grid: ',
num2str(iterations_gs)]);
```

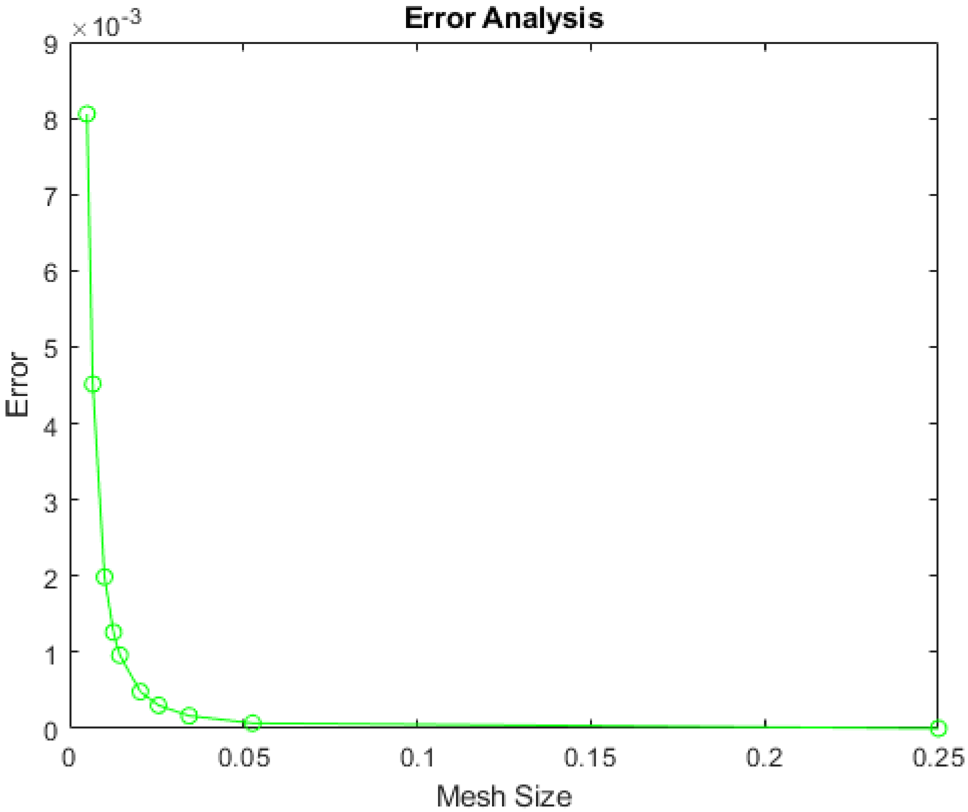
```
No. of Iterations in Gauss Seidel Method with Non-Uniform Grid: 727

No. of Iterations in Gauss Seidel Method with Uniform Grid: 523
```



Error Analysis

```
grid_points = [5, 20, 30, 40, 50, 70, 80, 100, 150, 200];

% Error (deviation from analytical solution) from the Gauss-Seidel method
Error_t = [];
mesh_size = [];

for grid_point = grid_points
    x_t = linspace(0, L, grid_point);
    mesh_size = [mesh_size, x_t(2) - x_t(1)];

    % Gauss Seidel
    T_t = zeros(1, grid_point);
    T_t(1) = Ta;
    T_t(grid_point) = Tb;

    T_old_t = T_t;
    iterations = 0;
    Error = 1;

    while Error > Tolerance
        for i = 2:grid_point-1
            T_t(i) = 0.5 * (T_t(i-1) + T_old_t(i+1));
        end

        Error = max(abs(T_t - T_old_t));
        iterations = iterations + 1;
        T_old_t = T_t;
    end

    % Analytical solution
    T_analytic = 100 * x_t + 300;
```

```
    % Norm 2 is used
    Error_t = [Error_t, (norm(T_t - T_analytic, 2) / norm(T_analytic, 2))];
end

% Plotting the results
figure;
plot(mesh_size, Error_t, 'g-o');
title("Error Analysis");
xlabel("Mesh Size");
ylabel("Error");
```



As the mesh size increases the error keeps on decreasing.

Part 2:

Consider a plane wall of thickness 2L (Fig. 2 with uniformly distributed heat source and, therefore, heat is generated at the rate q ‴ = 5 × 104 W/m3 . On each exposed surface, the wall is bounded by a circulating fluid of temperature T∞ = 25 °C. The convective heat trans�fer coefficient for both surfaces is h = 22 W/m2K. The thermal conductivity of the wall is given as k = 0.5 W/mK. Assuming the thickness of the wall is 50 cm and heat transfer is one-dimensional, solve for the temperature distribution across the wall numerically.

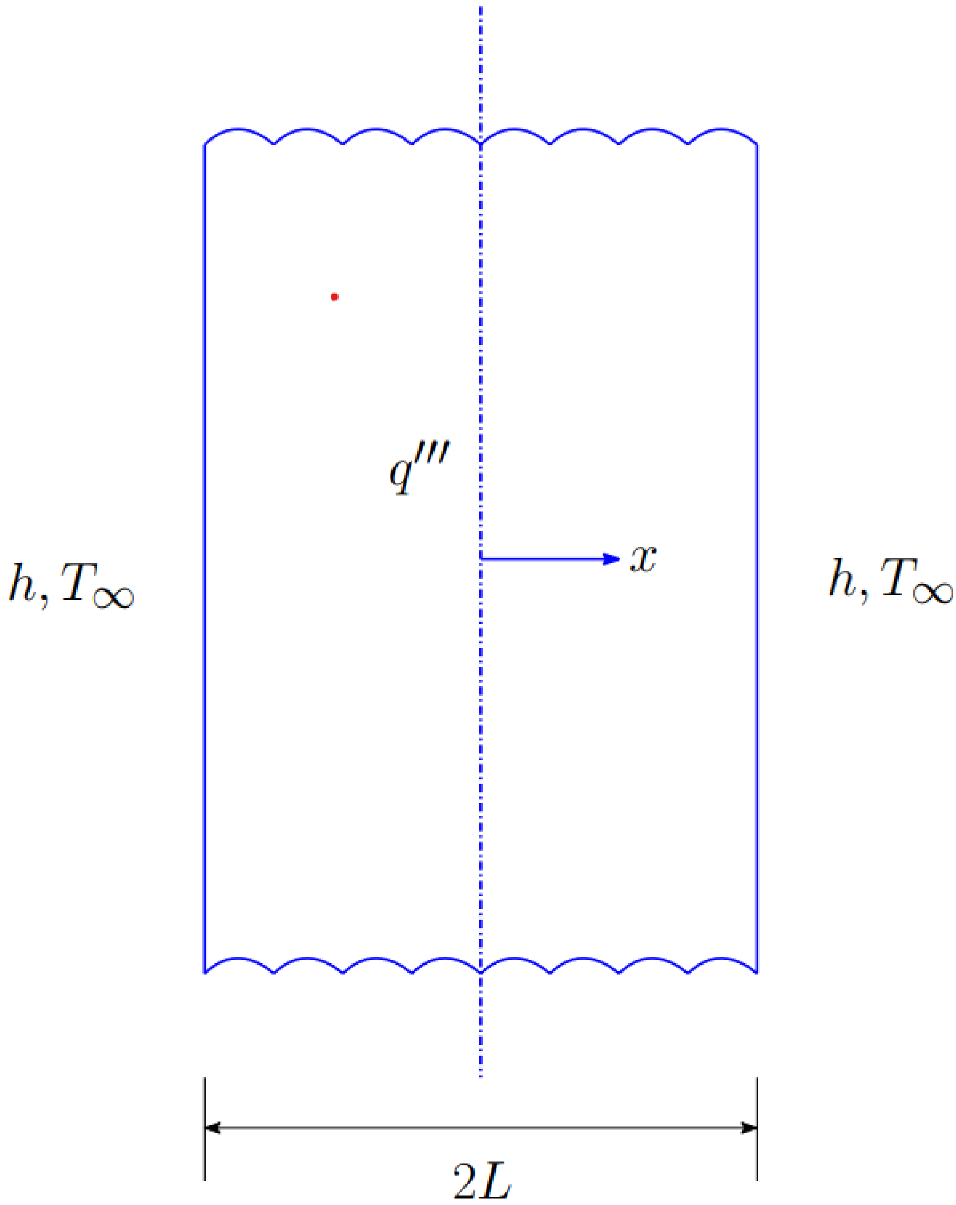
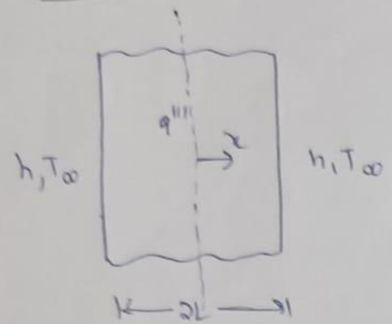


Figure 2: Schematic diagram of the slab

Question 2



Given, $T_\infty = 25^\circ C$

$q''' = 5 \times 10^4 \ W/m^3$

$h = 22 \ W/m^2K$

$k = 0.5 \ W/mK$

$2L = 50 cm$

Assuming 1D heat transfer $\rightarrow$

Governing eqn : $\dfrac{\partial^2 T}{\partial x^2} + \dfrac{q'''}{k} = 0$

Analytical $\displaystyle\int \frac{\partial^2 T}{\partial x^2} = \int -\frac{q'''}{k}$

$$T = -\frac{q''' x^2}{2k} + C_1 x + C_2$$

Also, $q''' = h \dfrac{\Delta T}{\Delta x} \Rightarrow T_{surf} = T_{amb} + q''' L / h$

@ $x = L$ $\quad T = T_{surf}$ $\quad \left[-k \dfrac{\partial T}{\partial x}\Big|_{x=L} = h (T_s - T_\infty) \right]$

@ $x = -L$ $\quad T = T_{surf}$

$$\Rightarrow T(x) = \frac{q''' L^2}{2k}\left(1 - \frac{x^2}{L^2}\right) + T_{surf}$$

Discretizing $\rightarrow$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2} = -\frac{q'''}{k}$$

$$\left\{\frac{1}{h^2}\right\} T_{i+1} + \left\{-\frac{2}{h^2}\right\} T_i + \left\{\frac{1}{h^2}\right\} T_{i-1} = -\frac{q'''}{k}$$

Let $c = +q''' h^2/k$

$$T_i = \frac{1}{2}\left[c + T_{i-1} + T_{i+1}\right]$$

Jacobi :- for k^{th} iteratn

$$T_i^k = \frac{1}{2}\left[c + T_{i-1}^{k-1} + T_{i+1}^{k-1}\right]$$

Gauss Siedel :-

$$T_i^k = \frac{1}{2}\left[c + T_{i-1}^{k} + T_{i+1}^{k-1}\right]$$

TDMA :-

$$\frac{2T_i}{h^2} = \frac{1}{h^2}T_{i-1} + \frac{1}{h^2}T_{i+1} + \frac{q'''}{k}$$

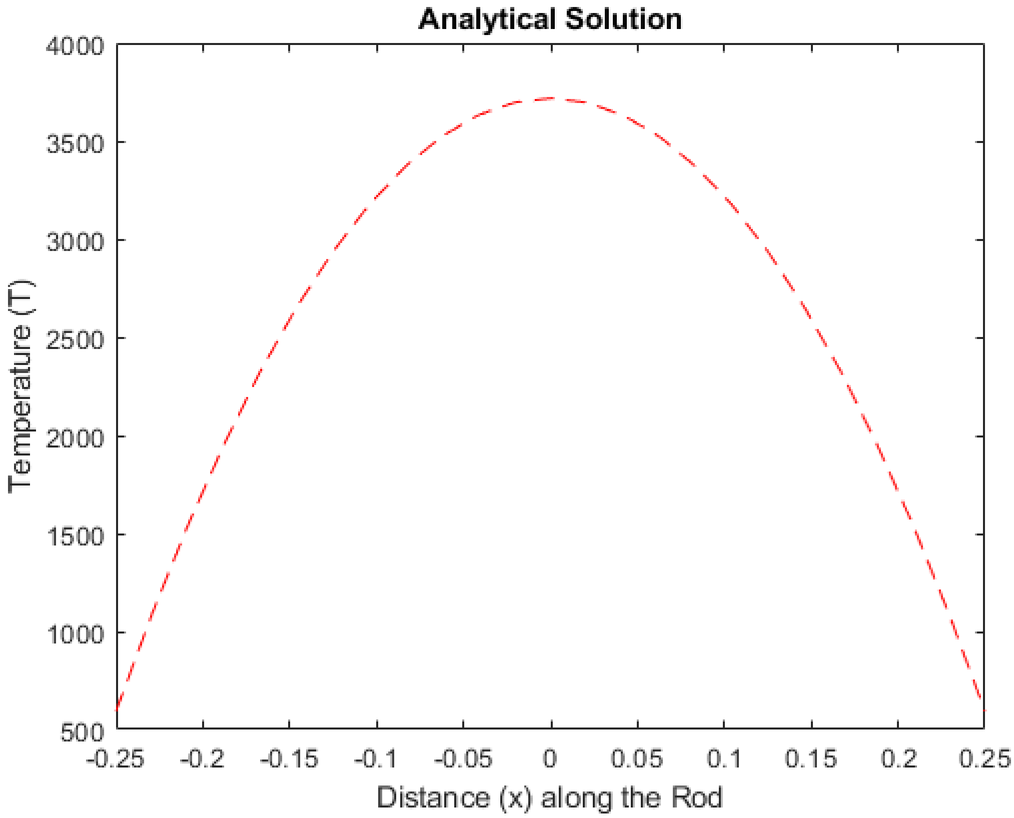
$$a = \frac{2}{h^2} \quad b = \frac{1}{h^2} \quad c = \frac{1}{h^2} \quad d = \frac{q'''}{k}$$

Code & plots:

Initial Conditions

```
% Abhishek Ghosh
% ME21BTECH11001

clc
clear all
% Given parameters
L = 0.5/2;
q_dot = 50000;
T_ambient = 25;
h_conv = 22;
```

```
k = 0.5;

n = 25;
Starting_coordinate = -L;
Ending_coordinate = L;

h = (Ending_coordinate - Starting_coordinate) / (n - 1);
x = linspace(-L, L, n);

% Analytical Solution
T_surf = T_ambient + q_dot * L / h_conv; % using Boundary Condition
T_analytical = 0.5 * q_dot * (L^2) * (1 - (x.^2)/(L^2)) / k + T_surf;

% Plotting
figure;
plot(x, T_analytical, 'r--');
title('Analytical Solution');
ylabel('Temperature (T)');
xlabel('Distance (x) along the Rod');
```

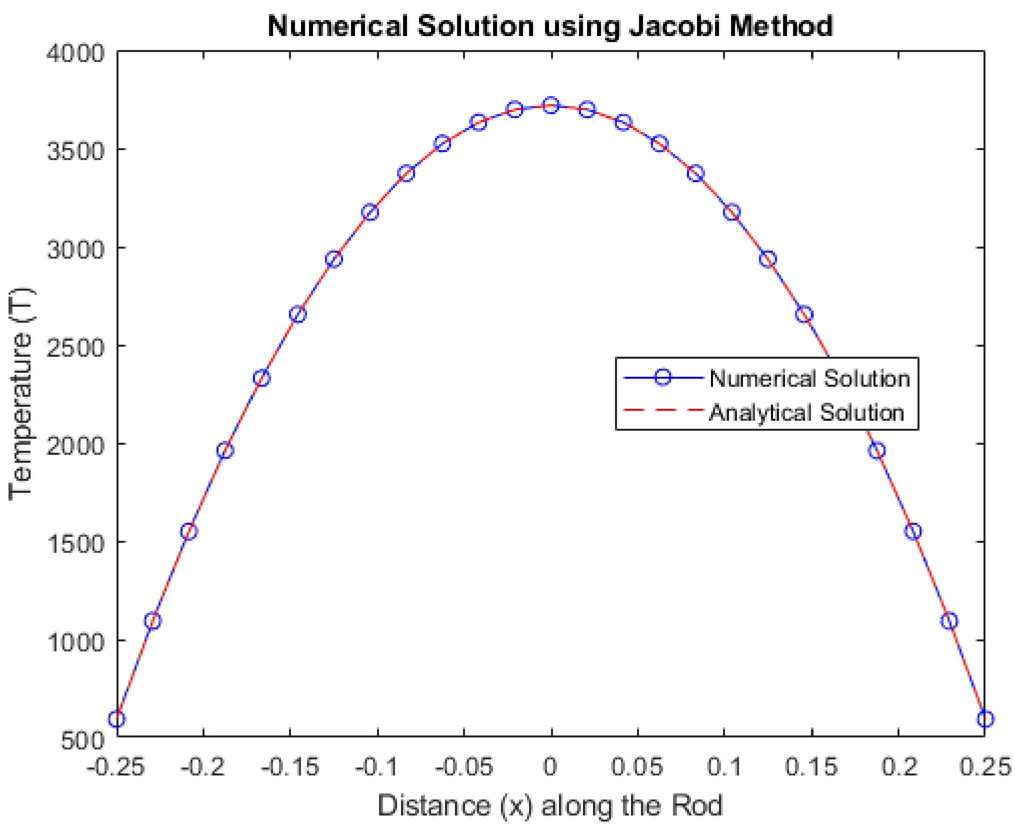


Jacobi Method

```
%Jacobi Method :-

% Initializing the temperature array
T_j = zeros(n, 1);

C = (q_dot * (h^2)) / k;

T_j(1) = T_surf;
T_j(n) = T_surf;

T_old_j = T_j;
iterations = 0;
Error = 1;
```

```
Tolerance = 1e-3;

while Error > Tolerance
    for i = 2:n-1
        T_j(i) = 0.5 * (C + T_old_j(i-1) + T_old_j(i+1));
    end

    Error = max(abs(T_j - T_old_j));
    T_old_j = T_j;
    iterations = iterations + 1;
end

% Plotting the results

figure;
plot(x, T_j, 'b-o');
hold on;
plot(x, T_analytical, 'r--');
title("Numerical Solution using Jacobi Method");
ylabel("Temperature (T)");
xlabel("Distance (x) along the Rod");
legend(["Numerical Solution", "Analytical Solution"], 'Location', 'Best');

disp(['No. of Iterations in Jacobi Method: ', num2str(iterations)]);
```

```
No. of Iterations in Jacobi Method: 1237
```

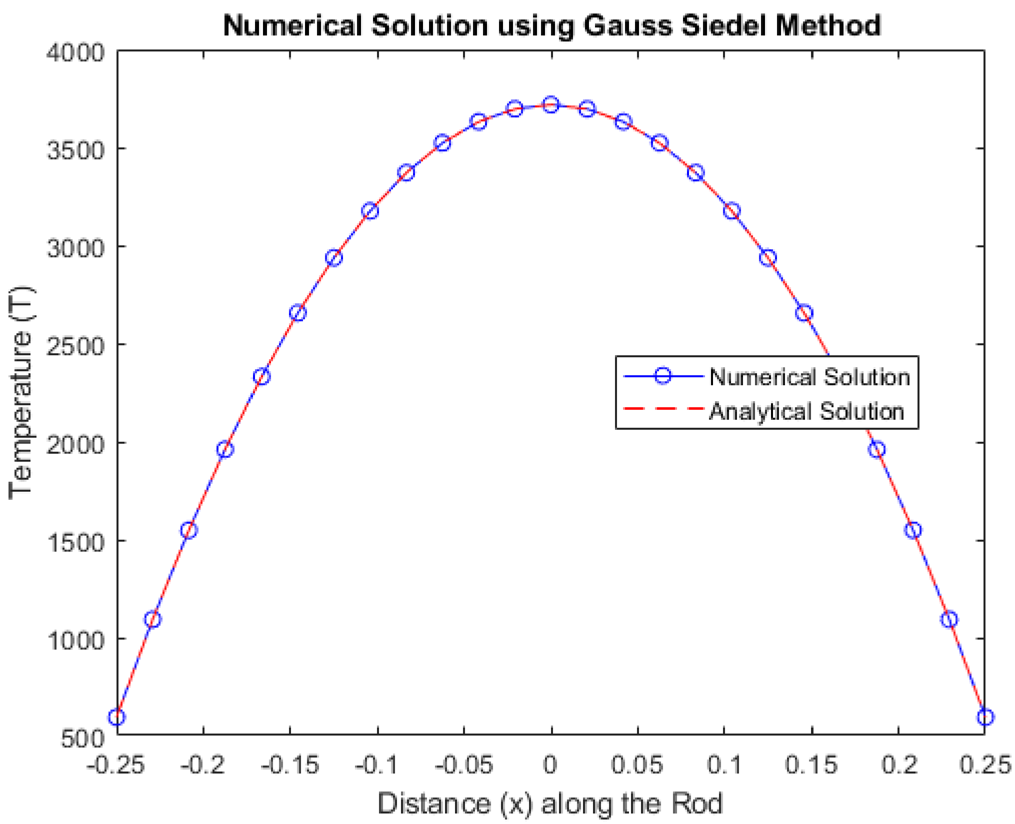


Gauss Siedel Method

```
% Gauss Siedel Method :-
```

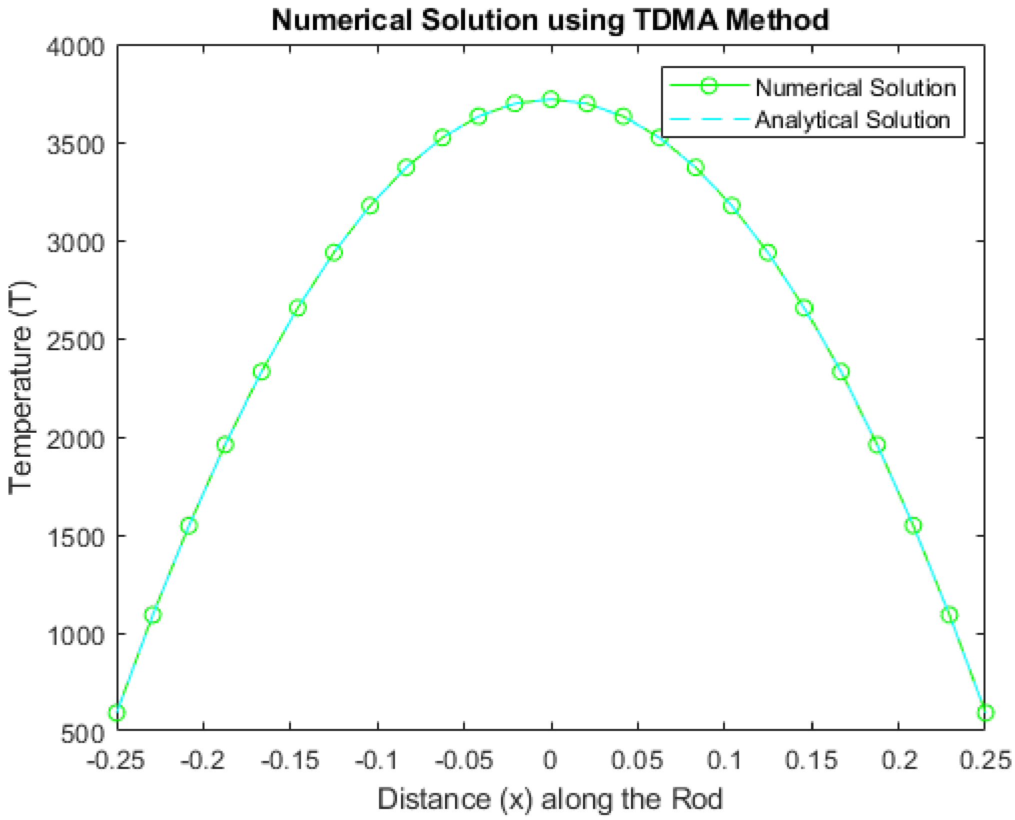
```
% Initializing the temperature array
T_j = zeros(n, 1);

C = (q_dot * (h^2)) / k;

T_j(1) = T_surf;
T_j(n) = T_surf;

T_old_j = T_j;
iterations = 0;
Error = 1;
Tolerance = 1e-3;

while Error > Tolerance
    for i = 2:n-1
        T_j(i) = 0.5 * (C + T_j(i-1) + T_old_j(i+1));
    end

    Error = max(abs(T_j - T_old_j));
    T_old_j = T_j;
    iterations = iterations + 1;
end

% Plotting the results

figure;
plot(x, T_j, 'b-o');
hold on;
plot(x, T_analytical, 'r--');
title("Numerical Solution using Gauss Siedel Method");
ylabel("Temperature (T)");
xlabel("Distance (x) along the Rod");
legend(["Numerical Solution", "Analytical Solution"], 'Location', 'Best');

disp(['No. of Iterations in Gauss Siedel Method: ', num2str(iterations)]);
```

```
No. of Iterations in Gauss Siedel Method: 649
```




```
% TDMA
T = zeros(n, 1);
T(1) = T_surf;
T(n) = T_surf;

P = zeros(n, 1);
Q = zeros(n, 1);

a = 2/h^2;
b = 1/h^2;
c = 1/h^2;
d = q_dot/k;

P(1) = 0;
Q(1) = T_surf;

% Forward sweep
for i = 2:n-1
    P(i) = b / (a - c*P(i-1));
    Q(i) = (d + c*Q(i-1)) / (a - c*P(i-1));
end

Q(n) = T(n);

% Backward substitution
for i = n-1:-1:2
    T(i) = T(i+1)*P(i) + Q(i);
end

figure;
plot(x, T, 'g-o');
hold on;
plot(x, T_analytical, 'c--');
title('Numerical Solution using TDMA Method');
ylabel('Temperature (T)');
```

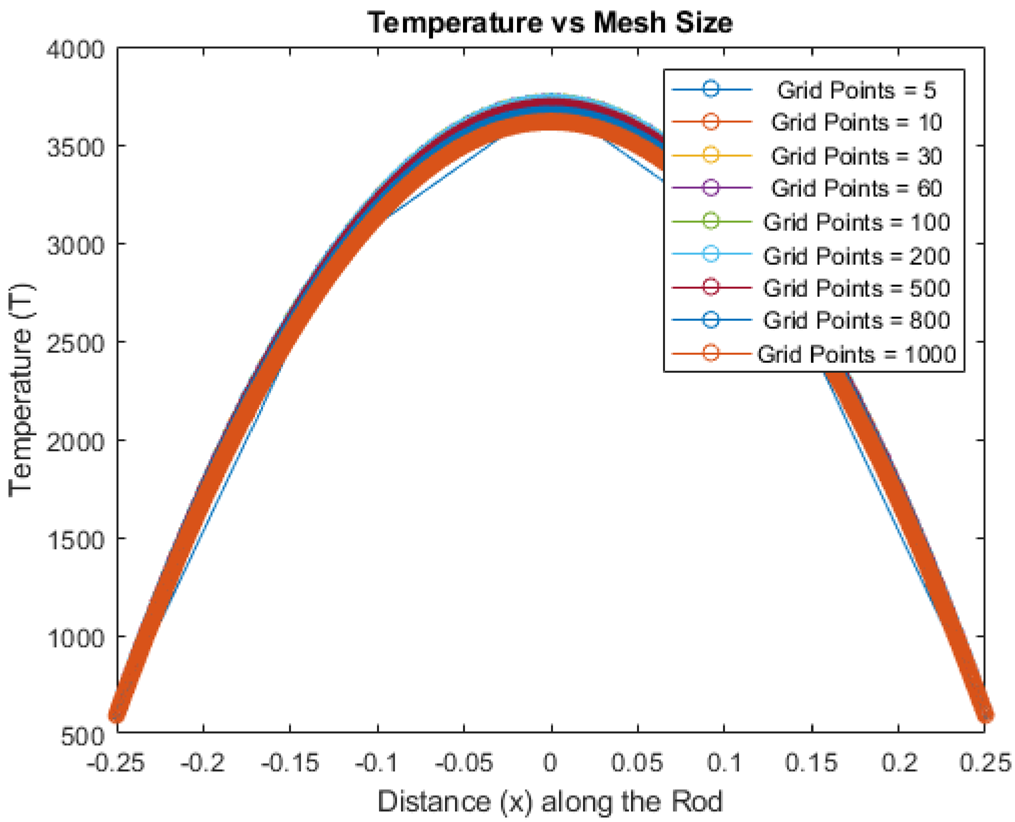
```
xlabel('Distance (x) along the Rod');
legend({'Numerical Solution', 'Analytical Solution'}, 'Location', 'northeast');
```



Mesh Independence

```
% Mesh independance

% Array of grid points
grid_points = [5, 10, 30, 60, 100, 200, 500, 800, 1000];
figure;
for m = grid_points
    % Initializing the temperature array
    T_j = zeros(m, 1);
    h = (Ending_coordinate - Starting_coordinate) / (m - 1);

    C = (q_dot * (h^2)) / k;

    T_j(1) = T_surf;
    T_j(m) = T_surf;

    T_old_j = T_j;
    iterations = 0;
    Error = 1;
    Tolerance = 1e-3;
    m

    while Error > Tolerance
        for i = 2:m-1
            T_j(i) = 0.5 * (C + T_j(i-1) + T_old_j(i+1));
        end
```

```
        Error = max(abs(T_j - T_old_j));
        T_old_j = T_j;
        iterations = iterations + 1;
    end

    % Plotting the results
    plot(linspace(-L, L, m), T_j, '-o');
    hold on;
end

title("Temperature vs Mesh Size");
ylabel("Temperature (T)");
xlabel("Distance (x) along the Rod");
legend(cellstr(num2str(grid_points', 'Grid Points = %d')));
hold off;
```



Increasing the number of mesh size would result in smoother curve that is it would take more iterations but would lead to more accurate results.