ME3180

FEM & CFD Theory

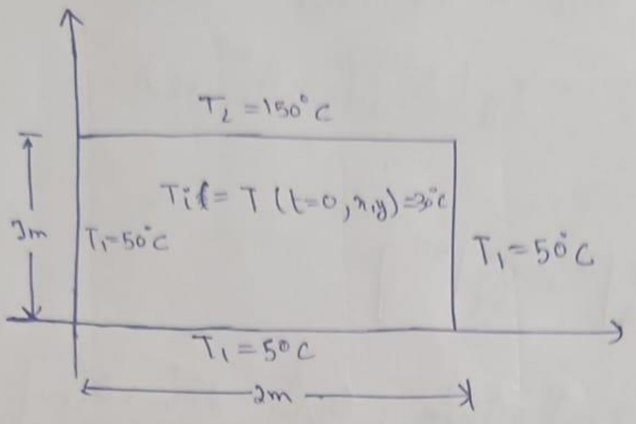
Assignment 3

Abhishek Ghosh

ME21BTECH11001

ME3180 Assignment 3

Abhishek Ghosh
ME21BTECH11001



$\alpha = 9.7 \times 10^{-5}\ m^3/s$

Governing eqtn $\dfrac{\partial T}{\partial t} = \alpha\left(\dfrac{\partial^2 T}{\partial x^2} + \dfrac{\partial^2 T}{\partial y^2}\right)$

Discretizing → ~~& using non-dimensional both~~

i) Explicit →

$$\frac{\theta_{i,j}^{n+1} - \theta_{i,j}^{n}}{\Delta t} = \alpha\left[\frac{\theta_{i-1,j}^{n} - 2\theta_{i,j}^{n} + \theta_{i+1,j}^{n}}{\Delta x^2} + \frac{\theta_{i,j-1}^{n} - 2\theta_{i,j}^{n} + \theta_{i,j+1}^{n}}{\Delta y^2}\right]$$

Considering $r_x = \dfrac{\alpha \Delta t}{\Delta x^2}$ & $r_y = \dfrac{\alpha \Delta t}{\Delta y^2}$

$$\theta_{i,j}^{n+1} = (1 - 2r_x - 2r_y)\theta_{i,j}^{n} + r_x\left(\theta_{i-1,j}^{n} + \theta_{i+1,j}^{n}\right) + r_y\left(\theta_{i,j-1}^{n} + \theta_{i,j+1}^{n}\right)$$

ii) Implicit →

$$\frac{\theta_{i,j}^{n+1} - \theta_{i,j}^{n}}{\Delta t} = \alpha\left[\frac{\theta_{i-1,j}^{n+1} - 2\theta_{i,j}^{n+1} + \theta_{i+1,j}^{n+1}}{\Delta x^2} + \frac{\theta_{i,j-1}^{n+1} - 2\theta_{i,j}^{n+1} + \theta_{i,j+1}^{n+1}}{\Delta y^2}\right]$$

$$(1+2r_x+2r_y)\,\theta_{i,j}^{n+1} = \theta_{i,j}^{n} + r_x\left(\theta_{i-1,j}^{n+1} + \theta_{i+1,j}^{n+1}\right)$$
$$+ r_y\left(\theta_{i,j-1}^{n+1} + \theta_{i,j+1}^{n+1}\right)$$

iii) Crank Nicholson :-

$$\frac{\partial\theta}{\partial t} = \frac{1}{2}\left[\left.\frac{\partial\theta}{\partial t}\right|^{n} + \left.\frac{\partial\theta}{\partial t}\right|^{n+1}\right]$$

$$\frac{\theta^{n+1}-\theta^{n}}{\Delta t} = \frac{\alpha}{2}\left[\frac{\theta_{i-1,j}^{n} - 2\theta_{i,j}^{n} + \theta_{i+1,j}^{n}}{\Delta x^2} + \frac{\theta_{i,j-1}^{n} - 2\theta_{i,j}^{n} + \theta_{i,j+1}^{n}}{\Delta y^2}\right]$$
$$+ \frac{\alpha}{2}\left[\frac{\theta_{i-1,j}^{n+1} - 2\theta_{i,j}^{n+1} + \theta_{i+1,j}^{n+1}}{\Delta x^2} + \frac{\theta_{i,j-1}^{n+1} - 2\theta_{i,j}^{n+1} + \theta_{i,j+1}^{n+1}}{\Delta y^2}\right]$$

$$\theta_{i,j}^{n+1} = \theta_{i,j}^{n} + \frac{r_1}{2}\left[\theta_{i-1,j}^{n} - 2\theta_{i,j}^{n} + \theta_{i+1,j}^{n}\right] + \frac{r_2}{2}\left[\theta_{i,j-1}^{n} - 2\theta_{i,j}^{n} + \theta_{i,j+1}^{n}\right]$$
$$+ \frac{r_1}{2}\left[\theta_{i-1,j}^{n+1} - 2\theta_{i,j}^{n+1} + \theta_{i+1,j}^{n+1}\right] + \frac{r_2}{2}\left[\theta_{i,j-1}^{n+1} - 2\theta_{i,j}^{n+1} + \theta_{i,j+1}^{n+1}\right]$$

$\cancel{(1+r_2)\theta_{i,j}^{n+1} = \frac{r_2}{2}\theta_{i,j+1}^{n+1} + \frac{r_2}{2}\theta_{i,j-1}^{n+1}}$ Apply TDMA after here

iv) ADI

(1) $$\left.\frac{\partial T}{\partial t}\right|^{*} = \alpha\left(\left.\frac{\partial^2 T}{\partial x^2}\right|^{*} + \left.\frac{\partial^2 T}{\partial y^2}\right|^{n}\right)$$

$$\Rightarrow \frac{T_{i,j}^{*} - T_{i,j}^{n}}{\Delta t/2} = \alpha\left\{\frac{T_{i+1,j}^{*} - 2T_{i,j}^{*} + T_{i-1,j}^{*}}{\Delta x^2} + \frac{T_{i,j-1}^{n} - 2T_{i,j}^{n} + T_{i,j+1}^{n}}{\Delta y^2}\right\}$$

$$\Rightarrow \left(1+\frac{\cancel{2}\alpha\Delta t}{(\Delta x)^2}\right)T^*_{i,j} - \frac{\alpha\Delta t}{2\Delta x^2}\left\{T^*_{i+1,j} + T^*_{i-1,j}\right\}$$

$$= \left(1-\frac{\cancel{2}\alpha\Delta t}{(\Delta y)^2}\right)T^n_{i,j} + \frac{\alpha\Delta t}{2(\Delta x)^2}\left\{T^n_{i,j-1} + T^n_{i,j+1}\right\}$$

(ii) $$\left.\frac{\partial T}{\partial t}\right|^{n+1} = \alpha\left(\left.\frac{\partial^2 T}{\partial x^2}\right|^{*} + \left.\frac{\partial^2 T}{\partial y^2}\right|^{n+1}\right)$$

$$\frac{T^{n+1}_{i,j} - T^*_{i,j}}{\Delta t/2} = \alpha\left\{\frac{T^*_{i+1,j} - 2T^*_{i,j} + T^*_{i-1,j}}{\Delta x^2} + \frac{T^{n+1}_{i,j-1} - 2T^{n+1}_{i,j} + T^{n+1}_{i,j+1}}{\Delta y^2}\right\}$$

$$\Rightarrow \left(1+\frac{\cancel{2}\alpha\Delta t}{\Delta y^2}\right)T^{n+1}_{i,j} - \frac{\alpha\Delta t}{2(\Delta y)^2}\left\{T^{n+1}_{i,j-1} + T^{n+1}_{i,j+1}\right\}$$

$$= \left(1-\frac{\cancel{2}\alpha\Delta t}{(\Delta x)^2}\right)T^*_{i,j} + \frac{\alpha\Delta t}{2(\Delta x)^2}\left(T^*_{i+1,j} + T^*_{i-1,j}\right)$$

Code & Results:

Note: *Different methods are commented out in the code*

Initial Conditions

```
% Abhishek Ghosh
% ME21BTECH11001

% Given Parameters
alpha = 9.7e-5;
T_init = 30;
T_1 = 50;
T_2 = 150;
L = 2; % length
B = 1; % breadth
nx = 31;
ny = 31;

x = linspace(0, L, nx);
y = linspace(0, B, ny);
dy = x(2) - x(1);
dx = y(2) - y(1);

% To find the maximum permissible time step, as per above set parameters
max_time = 10000;
delta_t = 0.5 / (alpha * ((1 / dx^2) + (1 / dy^2)));
disp(delta_t);
```

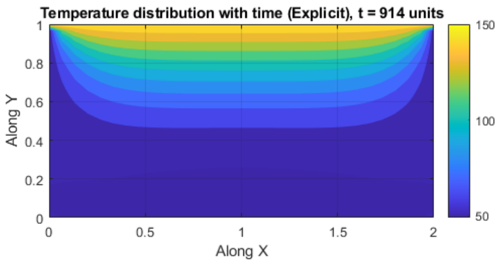
Explicit Method

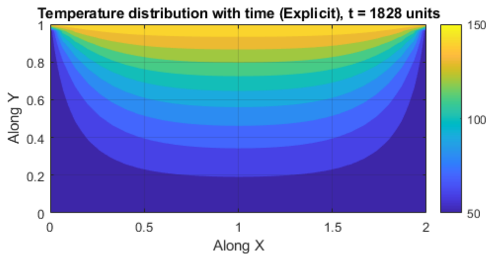
```
% Explicit
T_e = ones(nx, ny);

% Imposing Initial Conditions
T_e = T_init * T_e;

% Initializing Boundary Conditions
T_e(1, :) = T_1;
T_e(:, 1) = T_1;
T_e(:, ny) = T_1;
T_e(nx, :) = T_2;

t = 0.0; % time, initially it is 0
dt = 4.57; % time step

rx = alpha * dt / (dx^2);
ry = alpha * dt / (dy^2);

T_e_old = T_e;

iteration = 0;
time_intervals = 200;

while t <= max_time
    for i = 2:nx-1
        for j = 2:ny-1
            T_e(i, j) = rx * (T_e_old(i+1, j) + T_e_old(i-1, j)) + (1 - 2*rx -
2*ry) * T_e_old(i, j) + ry * (T_e_old(i, j+1) + T_e_old(i, j-1));
        end
    end

    t = t + dt;
    T_e_old = T_e;

    iteration = iteration + 1;

    % Plotting at specific time intervals
```

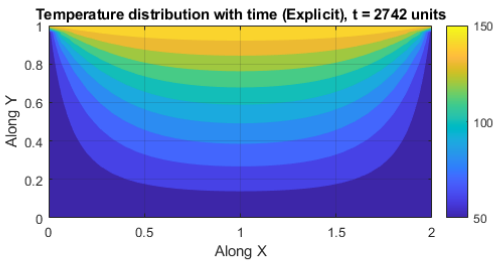
```
    if mod(iteration, time_intervals) == 0
        disp(['Number of iterations in Explicit Method: ', num2str(iteration)]);
        [X, Y] = meshgrid(x, y);
        figure;
        contourf(X, Y, T_e, 'LineColor', 'none');
        colorbar();
        title(['Temperature distribution with time (Explicit), t = ', num2str(t), '
units']);
        xlabel('Along X');
        ylabel('Along Y');
        axis equal;
        grid on;
    end
end

disp(['Number of iterations in Explicit Method: ', num2str(iteration)]);
```

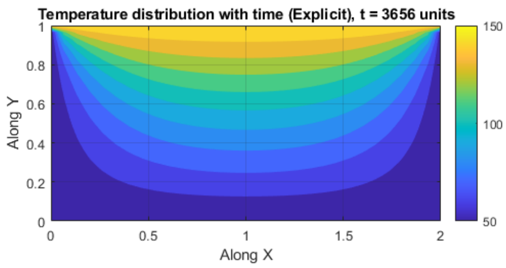
```
Number of iterations in Explicit Method: 2189
```

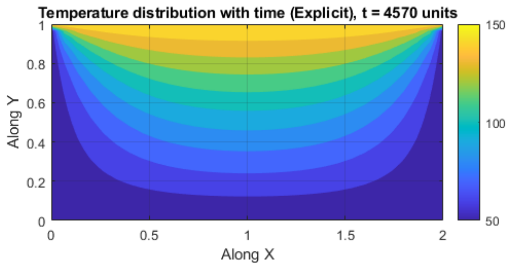
Temperature Contours at different time intervals for Explicit Method

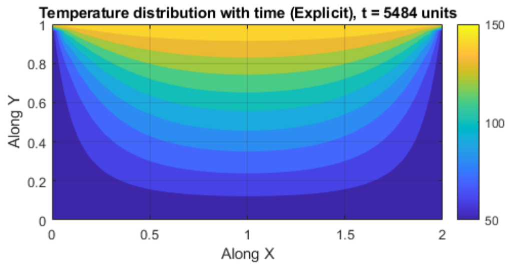


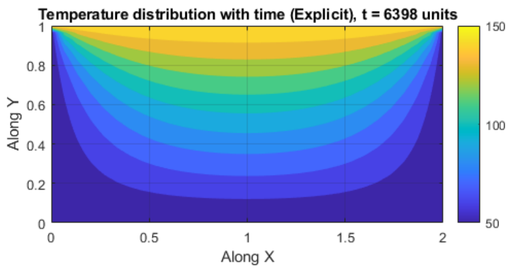


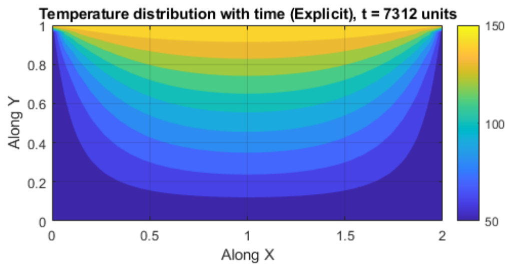


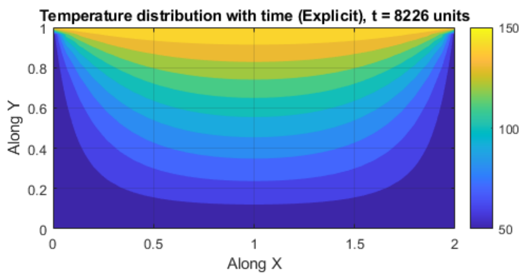


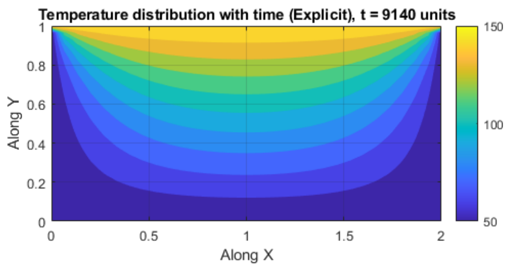












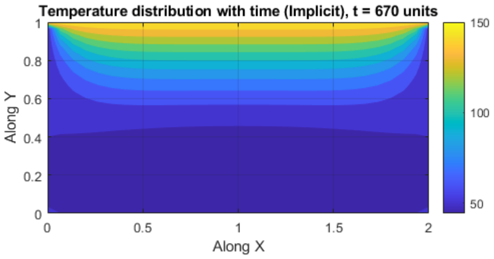
Implicit Scheme

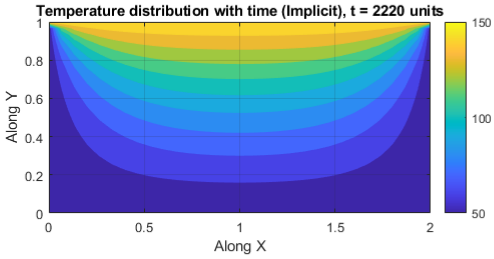
```
% Implicit Scheme
t = 0.0;
dt = 10; % Since Implicit Scheme is Unconditionally Stable, we can use any time
step
Tolerance = 1e-5;

rx = alpha * dt / (dx^2);
ry = alpha * dt / (dy^2);

T_i = ones(nx, ny) * T_init;

% Initializing Boundary Conditions
T_i(1, :) = T_1;
T_i(:, 1) = T_1;
```

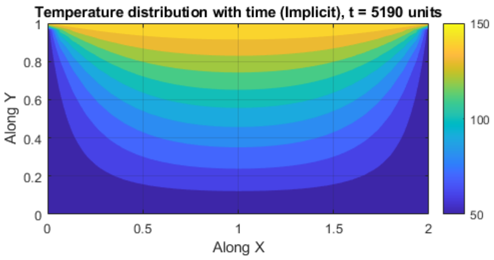
```
T_i(:, ny) = T_1;
T_i(nx, :) = T_2;

T_i_old = T_i;
T_prev = T_i;

iteration_count = 0;
time_intervals = 100;

while t <= max_time
    Error = 1;

    while Error > Tolerance
        for i = 2:nx-1
            for j = 2:ny-1
                T_i(i, j) = (T_prev(i,j) + rx*(T_i(i-1, j) + T_i(i+1, j)) +
ry*(T_i(i, j-1) + T_i(i, j+1))) / (1 + 2*rx + 2*ry);
            end
        end

        Error = max(max(abs(T_i - T_i_old)));
        T_i_old = T_i;

        iteration_count = iteration_count + 1;
    end

    if max(max(abs(T_i - T_prev))) < Tolerance
        break;
    end

    T_prev = T_i;
    t = t + dt;

    % Plotting at specific time intervals
    if mod(iteration_count, time_intervals) == 0
        disp(['Number of iterations in Implicit Method: ',
num2str(iteration_count)]);
        [X, Y] = meshgrid(x, y);
        figure;
        contourf(X, Y, T_i, 'LineColor', 'none');
        colorbar();
        title(['Temperature distribution with time (Implicit), t = ', num2str(t), '
units']);
        xlabel('Along X');
        ylabel('Along Y');
        axis equal;
        grid on;
    end
end

disp(['Number of total iterations in Implicit Method: ',
num2str(iteration_count)]);
```

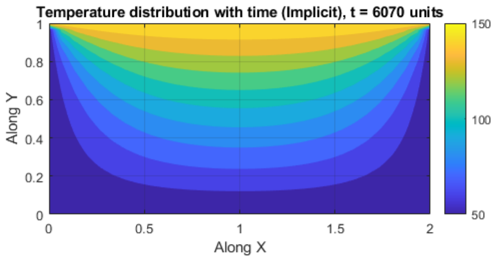
```
Number of total iterations in Implicit Method: 9168
```

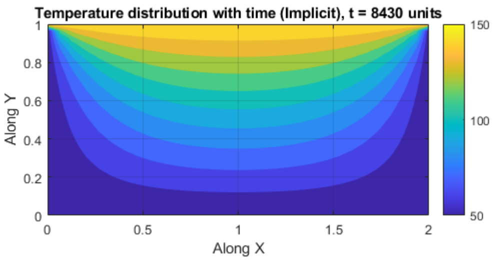
Temperature Contours at different time intervals for Implicit Method











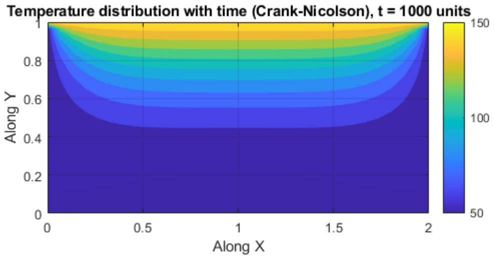
CN Method

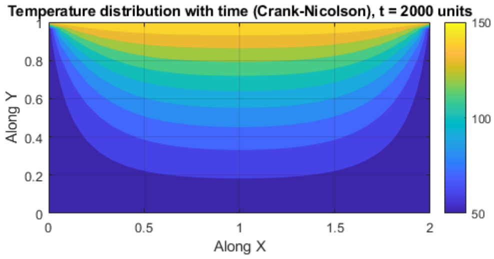
```
% CN Method
T_CN = ones(nx, ny);

t = 0.0;
dt = 1.0;

rx = alpha * dt / (dx^2);
ry = alpha * dt / (dy^2);

T_CN = T_CN * T_init;

% Initializing Boundary Conditions
T_CN(1, :) = T_1;
T_CN(:, 1) = T_1;
T_CN(:, ny) = T_1;
T_CN(nx, :) = T_2;
```

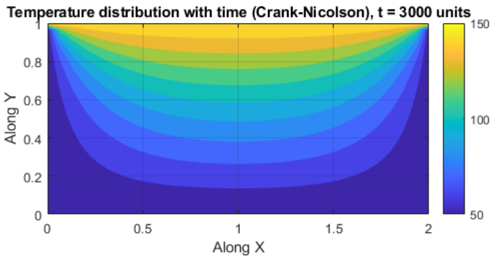
```
T_CN_old = T_CN;

iteration_count = 0;
time_intervals = 1000;

while t <= max_time
    a = (1 + ry + rx);
    b = 0.5 * ry;
    c = 0.5 * ry;
    T0 = T_1;
    Tn = T_1;

    for i = 2:nx-1
        d = zeros(1, ny);
        for j = 2:ny-1
            term1 = 0.5 * rx * (T_CN_old(i-1, j) - 2*T_CN_old(i, j) + T_CN_old(i+1,
j));
            term2 = 0.5 * ry * (T_CN_old(i, j-1) - 2*T_CN_old(i, j) + T_CN_old(i,
j+1));
            term3 = 0.5 * rx * (T_CN(i-1, j) + T_CN(i+1, j));
            d(j) = term1 + term2 + term3 + T_CN_old(i, j);
        end

        T_CN(i, :) = Thomas_algo(a, b, c, d, T0, Tn, ny);
    end

    if max(max(abs(T_CN - T_CN_old))) < Tolerance
        disp('Steady State Reached');
        break;
    end

    t = t + dt;
    T_CN_old = T_CN;

    iteration_count = iteration_count + 1;

    % Plotting at specific time intervals
    if mod(iteration_count, time_intervals) == 0
        disp(['Number of iterations in CN Method: ', num2str(iteration_count)]);
        [X, Y] = meshgrid(x, y);
        figure;
        contourf(X, Y, T_CN, 'LineColor', 'none');
        colorbar();
        title(['Temperature distribution with time (Crank-Nicolson), t = ',
num2str(t), ' units']);
        xlabel('Along X');
        ylabel('Along Y');
        axis equal;
        grid on;
    end
end

disp(['Number of total iterations in CN Method: ', num2str(iteration_count)]);
```

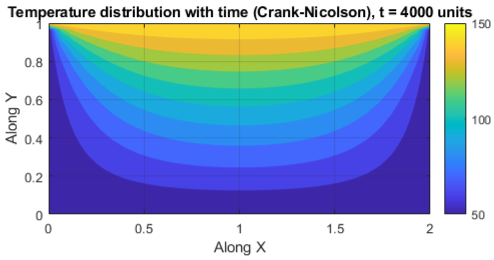
```
Number of total iterations in CN Method: 8133
```

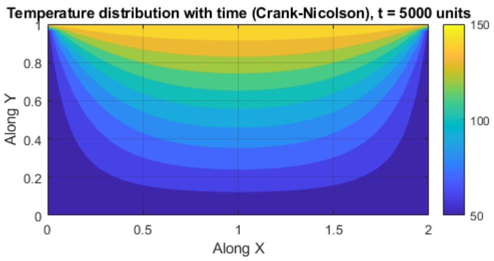
Temperature Contours at different time intervals for CN Method

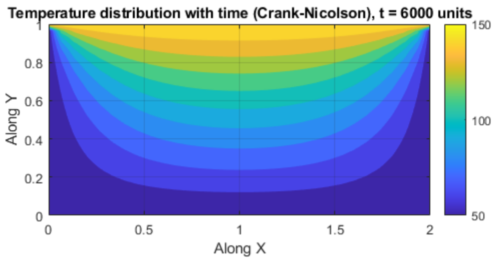

Temperature distribution with time (Crank-Nicolson), t = 1000 units
Along Y
Along X
150
100
50

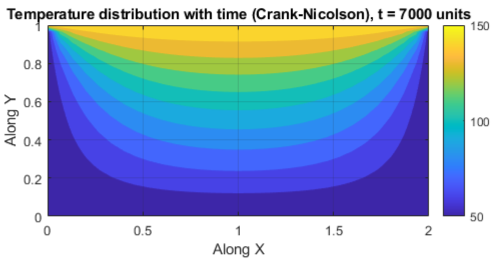

Temperature distribution with time (Crank-Nicolson), t = 2000 units
Along Y
Along X
150
100
50

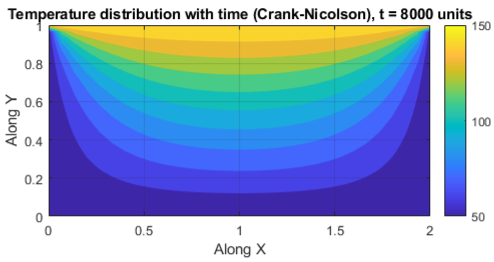

Temperature distribution with time (Crank-Nicolson), t = 3000 units
Along Y
Along X
150
100
50


Temperature distribution with time (Crank-Nicolson), t = 4000 units
Along Y
Along X
150
100
50


Temperature distribution with time (Crank-Nicolson), t = 5000 units
Along Y
Along X
150
100
50


Temperature distribution with time (Crank-Nicolson), t = 6000 units
Along Y
Along X
150
100
50


Temperature distribution with time (Crank-Nicolson), t = 7000 units
Along Y
Along X
150
100
50


Temperature distribution with time (Crank-Nicolson), t = 8000 units
Along Y
Along X
150
100
50

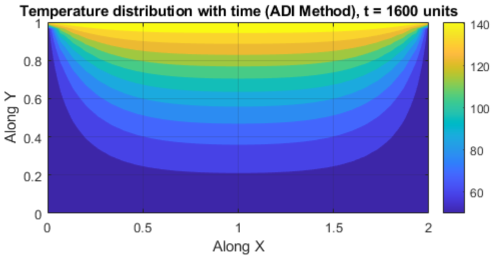
ADI Method

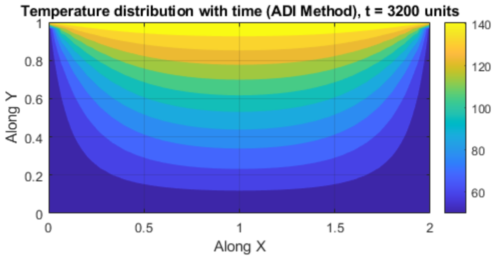
```
% ADI
t = 0.0;
dt = 8;

rx = alpha * dt / (dx^2);
ry = alpha * dt / (dy^2);

T_adt = ones(nx, ny) * T_init;

% Initializing Boundary Conditions
T_adt(1, :) = T_1;
T_adt(:, 1) = T_1;
T_adt(:, ny) = T_1;
T_adt(nx, :) = T_2;

T_adt_old = T_adt;
iteration_count = 0;
time_intervals = 200;

while t <= max_time
    Temp = T_adt_old;

    % Sweeping in X - Direction
    a = (1 + ry);
    b = 0.5 * ry;
    c = 0.5 * ry;

    T0 = T_1;
    Tn = T_1;

    for i = 2:nx-1
        d = zeros(1, ny);
        for j = 2:ny-1
            d(j) = 0.5 * rx * T_adt_old(i-1, j) + (1 - rx) * T_adt_old(i, j) + 0.5
* rx * T_adt_old(i+1, j);
        end

        Temp(i, :) = Thomas_algo(a, b, c, d, T0, Tn, ny);
    end

    T_adt = Temp;

    % Sweeping in Y Direction
    a = 1 + rx;
    b = 0.5 * rx;
    c = 0.5 * rx;

    T0 = T_1;
    Tn = T_2;

    for j = 2:ny-1
        d = zeros(1, nx);
        for i = 2:nx-1
            d(i) = 0.5 * ry * Temp(i, j-1) + (1 - ry) * Temp(i, j) + 0.5 * ry *
Temp(i, j+1);
        end

        T_adt(:, j) = Thomas_algo(a, b, c, d, T0, Tn, nx);
    end

    if max(max(abs(T_adt - T_adt_old))) < Tolerance
        disp('Steady State Reached');
        break;
    end
```

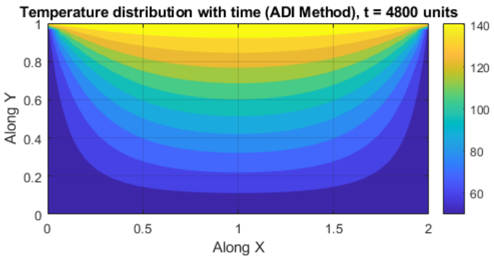
```
    t = t + dt;
    T_adt_old = T_adt;
    iteration_count = iteration_count + 1;

    % Plotting at specific time intervals
    if mod(iteration_count, time_intervals) == 0
        disp(['Number of iterations in ADI Method: ', num2str(iteration_count)]);
        [X, Y] = meshgrid(x, y);
        figure;
        contourf(X, Y, T_adt, 10, 'LineColor', 'none');
        colorbar();
        title(['Temperature distribution with time (ADI Method), t = ', num2str(t),
' units']);
        xlabel('Along X');
        ylabel('Along Y');
        axis equal;
        grid on;
    end
end

disp(['Number of total iterations in ADI Method: ', num2str(iteration_count)]);



% TDMA Function
function T_tdma = Thomas_algo(a, b, c, d, T0, Tn, n)
    T_tdma = zeros(1, n);
    T_tdma(1) = T0;
    T_tdma(n) = Tn;

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

    P(1) = 0;
    Q(1) = T0;

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

    Q(n) = Tn;

    for i = n-1:-1:1
        T_tdma(i) = T_tdma(i+1) * P(i) + Q(i);
    end
end
```

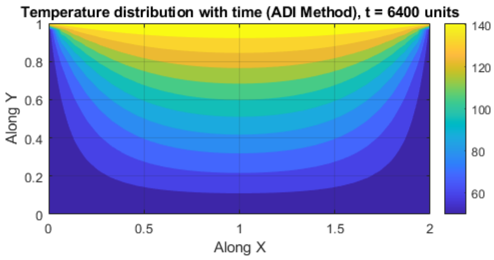
```
Number of total iterations in ADI Method: 1196
```

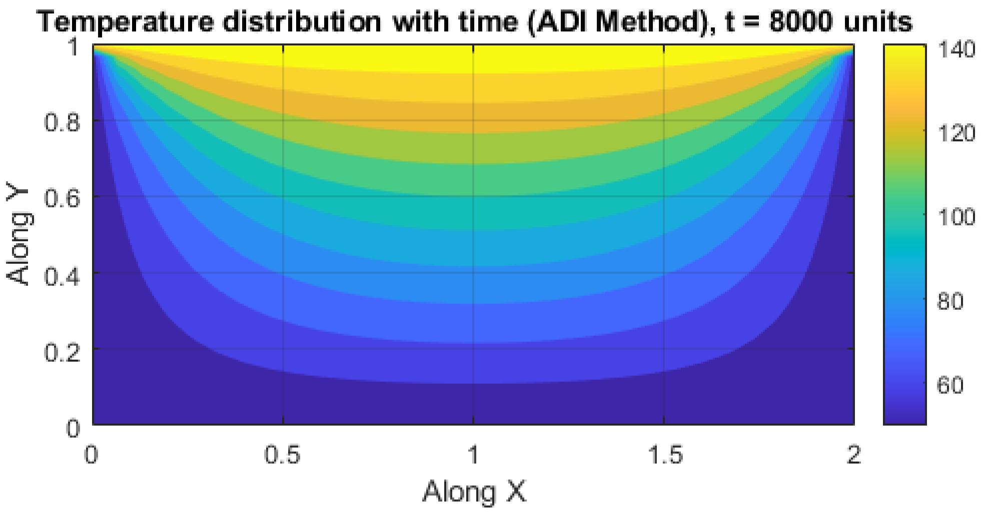
Temperature Contours at different time intervals for ADI Method


Temperature distribution with time (ADI Method), t = 1600 units
Along Y
Along X
0
0.2
0.4
0.6
0.8
1
0.5
1.5
2
140
120
100
80
60


Temperature distribution with time (ADI Method), t = 3200 units
Along Y
Along X
0
0.2
0.4
0.6
0.8
1
0.5
1.5
2
140
120
100
80
60


Temperature distribution with time (ADI Method), t = 4800 units
Along Y
Along X
0
0.2
0.4
0.6
0.8
1
0.5
1.5
2
140
120
100
80
60

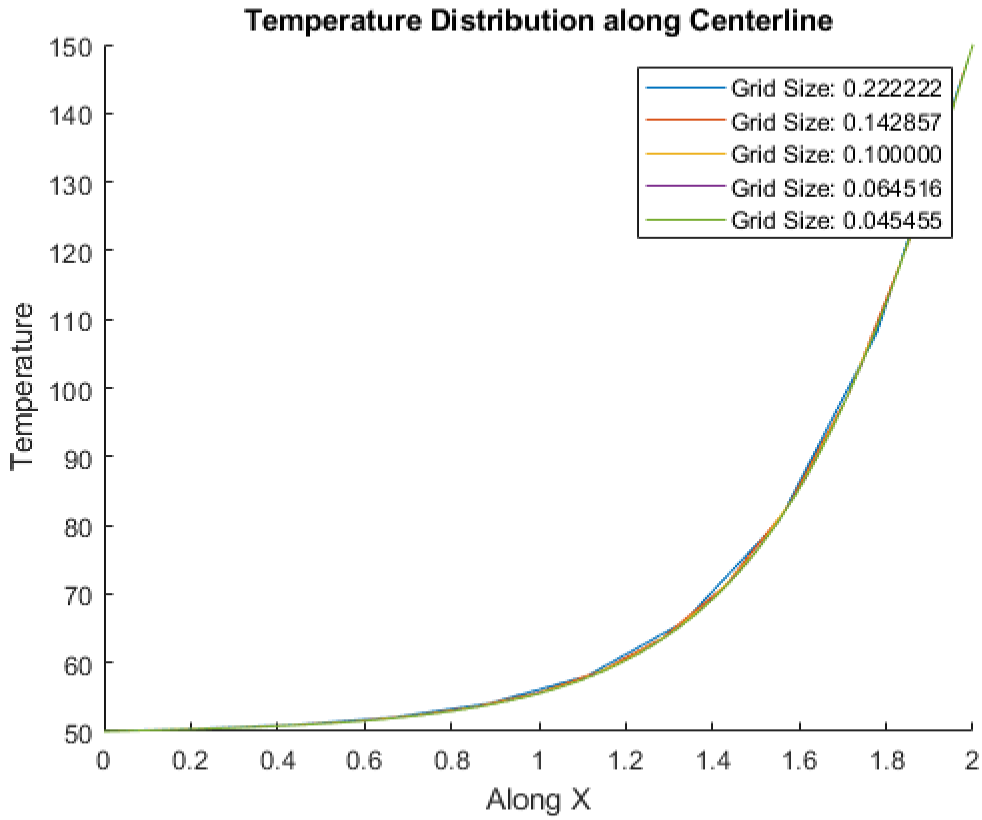

Temperature distribution with time (ADI Method), t = 6400 units
Along Y
Along X
0
0.2
0.4
0.6
0.8
1
0.5
1.5
2
140
120
100
80
60


Temperature distribution with time (ADI Method), t = 8000 units
Along Y
Along X
0
0.2
0.4
0.6
0.8
1
0.5
1.5
2
140
120
100
80
60

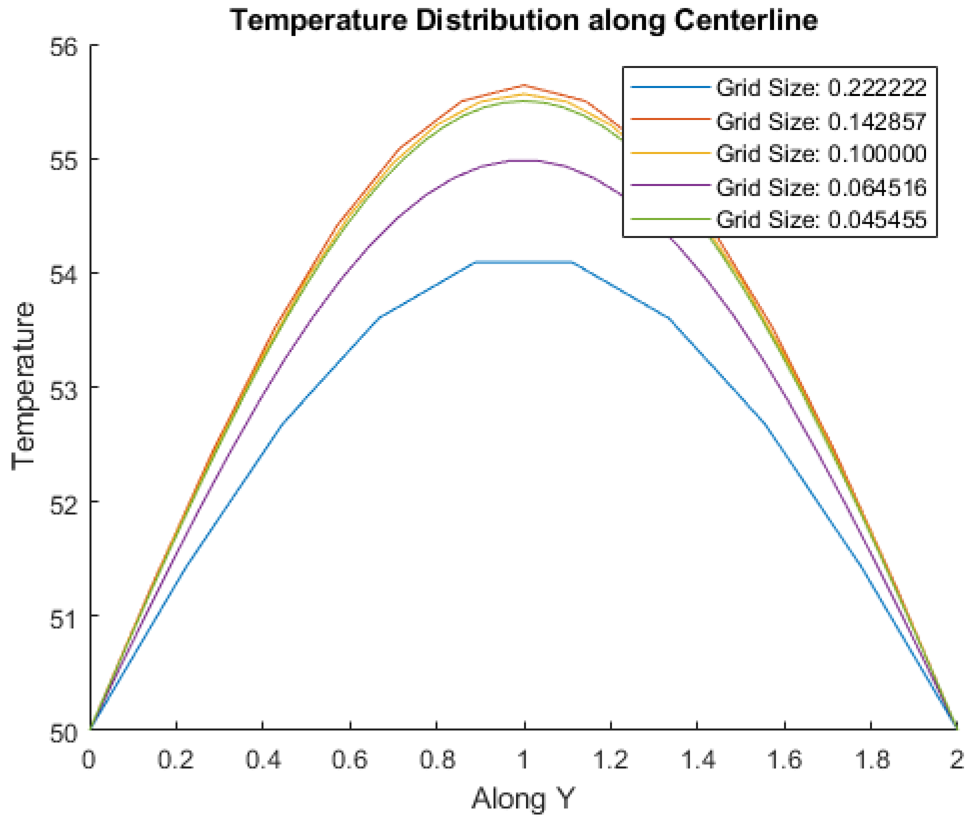
Grid Independence (Implicit Scheme)

```
grid_points = [10, 15, 21, 32, 45];



figure;

hold on;
for g = grid_points
    x_t = linspace(0, L, g);
    y_t = linspace(0, B, g);
    dx = x_t(2) - x_t(1);
    dy = y_t(2) - y_t(1);

    T = implicit(g);
    T_plot = T(:, round(g/2)); % Corrected indexing


    plot(x_t, T_plot, 'DisplayName', sprintf('Grid Size: %.6f', x_t(2) - x_t(1)));

end

hold off;
title('Temperature Distribution along Centerline');
xlabel('Along X');
ylabel('Temperature');
legend;

figure;
hold on;
for g = grid_points
    x_t = linspace(0, L, g);
    y_t = linspace(0, B, g);
    dx = x_t(2) - x_t(1);
    dy = y_t(2) - y_t(1);

    T = implicit(g);
    T_plot = T( round(g/2),:); % Corrected indexing


    plot(x_t, T_plot, 'DisplayName', sprintf('Grid Size: %.6f', x_t(2) - x_t(1)));

end

hold off;
title('Temperature Distribution along Centerline');
xlabel('Along Y');
ylabel('Temperature');
legend;

function T_i = implicit(grid_size)
    alpha = 9.7e-5;
    T_init = 30;
    T_1 = 50;
    T_2 = 150;
    L = 2; % length
    B = 1; % breadth
    % Initialize grid and parameters based on the grid size
    nx = grid_size;
    ny = grid_size;
    dx = L / (nx - 1);
    dy = B / (ny - 1);
```

```
    % Initialize temperature field and boundary conditions
    T_i = ones(nx, ny) * T_init;
    T_i(1, :) = T_1;
    T_i(:, 1) = T_1;
    T_i(:, ny) = T_1;
    T_i(nx, :) = T_2;
    max_time = 10000;
    T_i_old = T_i;
    t = 0.0;
    dt = 10; % Adjust the time step as needed
    Tolerance = 1e-5;
    rx = alpha * dt / (dx^2);
    ry = alpha * dt / (dy^2);
    iteration_count = 0;

    % Perform the implicit scheme simulation
    while t <= max_time
        Error = 1;

        while Error > Tolerance
            for i = 2:nx-1
                for j = 2:ny-1
                    T_i(i, j) = (T_i_old(i, j) + rx * (T_i(i-1, j) + T_i(i+1, j)) +
ry * (T_i(i, j-1) + T_i(i, j+1))) / (1 + 2 * rx + 2 * ry);
                end
            end

            Error = max(max(abs(T_i - T_i_old)));
            T_i_old = T_i;

            iteration_count = iteration_count + 1;
        end

        if max(max(abs(T_i - T_i_old))) < Tolerance
            break;
        end

        t = t + dt;
        T_i_old = T_i;
    end
end
```



Temperature Distribution along Centerline
150
140
130
120
110
100
90
80
70
60
50
Temperature
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Along X
Grid Size: 0.222222
Grid Size: 0.142857
Grid Size: 0.100000
Grid Size: 0.064516
Grid Size: 0.045455



Temperature Distribution along Centerline
56
55
54
53
52
51
50
Temperature
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Along Y
Grid Size: 0.222222
Grid Size: 0.142857
Grid Size: 0.100000
Grid Size: 0.064516
Grid Size: 0.045455