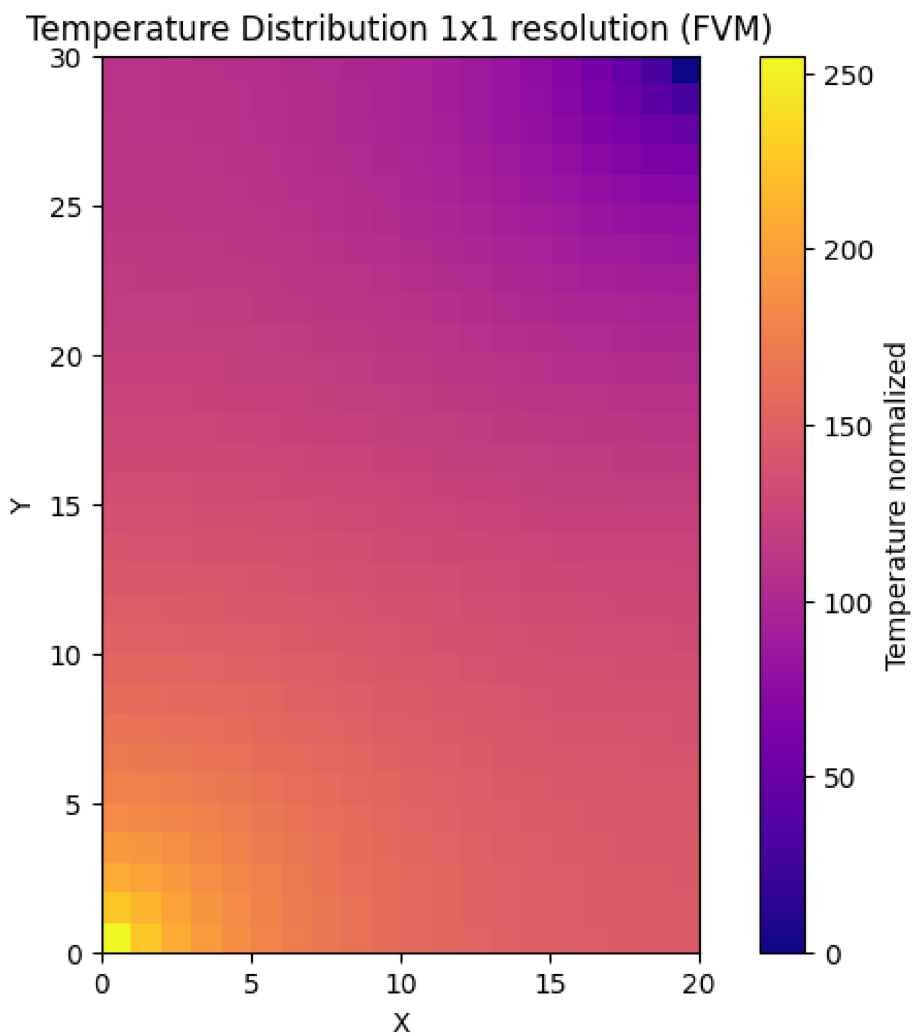
```python
import numpy as np
import matplotlib.pyplot as plt

# Helper function for 1D indexing
def idx(i, j, grid_height):
    return i * grid_height + j

def calculate_grid_cells(domain_width, domain_height, step_x, step_y):
    cells_x = int(domain_width // step_x)
    cells_y = int(domain_height // step_y)
    return cells_x, cells_y

def solve_heat_fvm(cells_x, cells_y, thermal_conductivity, heat_flux,
step_x, step_y):
    total_cells = cells_x * cells_y
    coefficient_matrix = np.zeros((total_cells, total_cells))
    load_vector = np.zeros(total_cells)

    for row in range(cells_x):
        for col in range(cells_y):
            cell_index = idx(row, col, cells_y)

            coef_upper = thermal_conductivity / (step_y ** 2) if row >
0 else 0
            coef_right = thermal_conductivity / (step_x ** 2) if col <
cells_y - 1 else 0
            coef_lower = thermal_conductivity / (step_y ** 2) if row <
cells_x - 1 else 0
            coef_left = thermal_conductivity / (step_x ** 2) if col >
0 else 0
            coef_diag = -(coef_upper + coef_right + coef_lower +
coef_left)

            coefficient_matrix[cell_index, cell_index] = coef_diag
            if row > 0:
                coefficient_matrix[cell_index, idx(row - 1, col,
cells_y)] = coef_upper
            if col < cells_y - 1:
                coefficient_matrix[cell_index, idx(row, col + 1,
cells_y)] = coef_right
            if row < cells_x - 1:
                coefficient_matrix[cell_index, idx(row + 1, col,
cells_y)] = coef_lower
            if col > 0:
                coefficient_matrix[cell_index, idx(row, col - 1,
cells_y)] = coef_left

    heat_source_index = idx(cells_x - 1, 0, cells_y)
    heat_sink_index = idx(0, cells_y - 1, cells_y)
```

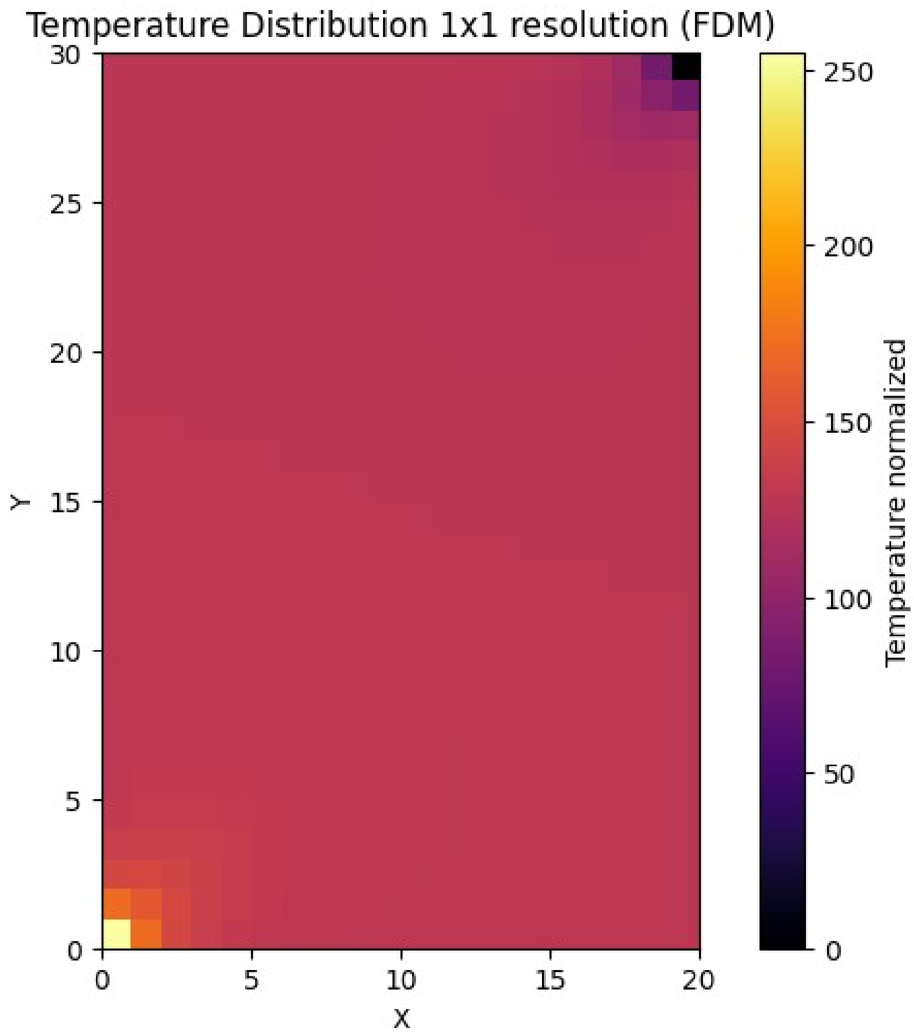
```
    load_vector[heat_source_index] = heat_flux
    load_vector[heat_sink_index] = -heat_flux

    ground_index = idx(0, 0, cells_y)
    coefficient_matrix = np.delete(coefficient_matrix, ground_index,
axis=0)
    coefficient_matrix = np.delete(coefficient_matrix, ground_index,
axis=1)
    load_vector = np.delete(load_vector, ground_index)

    load_vector = -load_vector
    temperature_vector = np.linalg.solve(coefficient_matrix,
load_vector)
    temperature_vector = np.insert(temperature_vector, ground_index,
0)
    temperature_field = temperature_vector.reshape((cells_x, cells_y))

    return temperature_field, coefficient_matrix, load_vector

def create_fd_coeff_matrix(cells_x, cells_y, heat_flux):
    total_nodes = cells_x * cells_y
    matrix_a = np.eye(total_nodes) * 4
    matrix_a += np.eye(total_nodes, k=1) * -1
    matrix_a += np.eye(total_nodes, k=-1) * -1
    matrix_a += np.eye(total_nodes, k=cells_y) * -1
    matrix_a += np.eye(total_nodes, k=-cells_y) * -1

    rhs_vector = np.zeros(total_nodes)

    for row in range(cells_x):
        if row != cells_x - 1:
            row_end = (row + 1) * cells_y - 1
            matrix_a[row_end, row_end + 1] = 0
            matrix_a[row_end + 1, row_end] = 0

    heat_source_index = idx(cells_x - 1, 0, cells_y)
    heat_sink_index = idx(0, cells_y - 1, cells_y)

    rhs_vector[heat_source_index] = heat_flux
    rhs_vector[heat_sink_index] = -heat_flux

    ground_node_index = 0
    matrix_a[ground_node_index, :] = 0
    matrix_a[ground_node_index, ground_node_index] = 1
    rhs_vector[ground_node_index] = 0

    return matrix_a, rhs_vector

def solve_heat_fdm(cells_x, cells_y, thermal_conductivity, heat_flux,
step_x, step_y):
```

```
    coeff_matrix, rhs_vector = create_fd_coeff_matrix(cells_x,
cells_y, heat_flux)
    temperature_vector = np.linalg.solve(coeff_matrix, rhs_vector)
    temperature_field = temperature_vector.reshape((cells_x, cells_y))
    temperature_field += 25
    return temperature_field

def plot_heatmap(temperature_field, step_x, step_y, title="Temperature
Distribution", origin="upper", cmap="coolwarm"):
    normalized_temp = (temperature_field - np.min(temperature_field))
/ (np.max(temperature_field) - np.min(temperature_field)) * 255
    plt.figure(figsize=(8, 6))
    extent = [0, temperature_field.shape[1] * step_x, 0,
temperature_field.shape[0] * step_y]
    plt.imshow(normalized_temp, cmap=cmap, origin=origin,
extent=extent)
    plt.colorbar(label="Temperature normalized")
    plt.title(title)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()

def plot_temp_profile_x(temperature_field, step_x, resolution=(1,1),
method="FVM"):
    title = f"Temperature Distribution Along X-axis
({resolution[0]}x{resolution[1]} resolution {method})"
    domain_center_x = (temperature_field.shape[1] * step_x) / 2
    x_positions = np.linspace(-domain_center_x, domain_center_x,
temperature_field.shape[1])
    mid_row_values = temperature_field[temperature_field.shape[0] //
2, :]
    plt.figure(figsize=(10, 6))
    plt.plot(x_positions, mid_row_values, 'o-', label="Along X-axis")
    plt.title(title)
    plt.xlabel("Position along X-axis (centered at 0)")
    plt.ylabel("Temperature")
    plt.grid(True)
    plt.legend()
    plt.show()

def plot_temp_profile_y(temperature_field, step_y, resolution=(1,1),
method="FVM"):
    title = f"Temperature Distribution Along Y-axis
({resolution[0]}x{resolution[1]} resolution {method})"
    domain_center_y = (temperature_field.shape[0] * step_y) / 2
    y_positions = np.linspace(-domain_center_y, domain_center_y,
temperature_field.shape[0])
    mid_column_values = temperature_field[:,
temperature_field.shape[1] // 2]
    mid_column_values = mid_column_values[::-1]
```

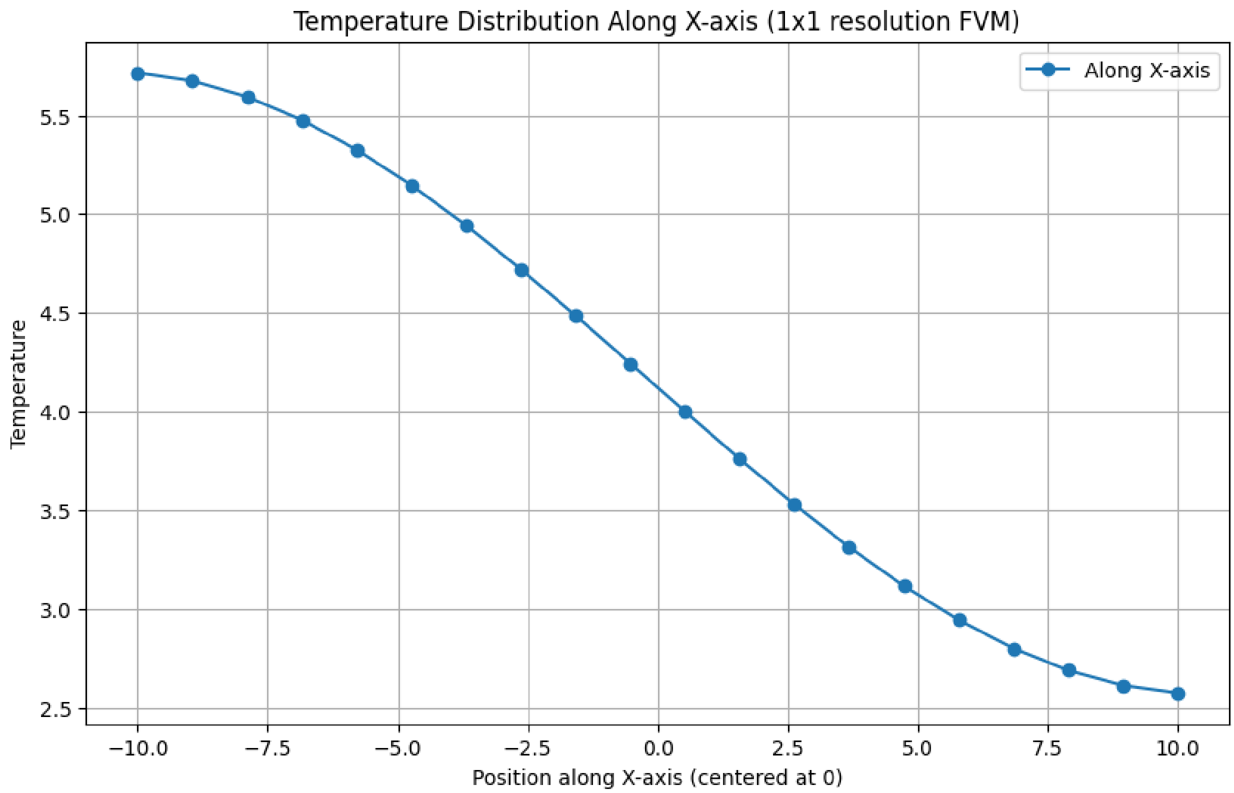
```
    plt.figure(figsize=(10, 6))
    plt.plot(y_positions, mid_column_values, 's-', label="Along Y-axis
(X=0)")
    plt.title(title)
    plt.xlabel("Position along Y-axis (centered at 0)")
    plt.ylabel("Temperature")
    plt.grid(True)
    plt.legend()
    plt.show()

def main():
    domain_width, domain_height = 30, 20
    thermal_conductivity = 390
    heat_flux = 5000

    step_x, step_y = 1, 1
    cells_x, cells_y = calculate_grid_cells(domain_width,
domain_height, step_x, step_y)
    temperature_fvm, coeff_matrix, load_vector =
solve_heat_fvm(cells_x, cells_y, thermal_conductivity, heat_flux,
step_x, step_y)
    temperature_fdm = solve_heat_fdm(cells_x, cells_y,
thermal_conductivity, heat_flux, step_x, step_y)

    plot_heatmap(temperature_fvm, step_x, step_y, title="Temperature
Distribution "f'{step_x}x{step_y}'" resolution (FVM)", cmap="plasma")
    plot_heatmap(temperature_fdm, step_x, step_y, title="Temperature
Distribution "f'{step_x}x{step_y}'" resolution (FDM)", cmap="inferno")

    plot_temp_profile_x(temperature_fvm, step_x, (step_x, step_y),
"FVM")
    plot_temp_profile_y(temperature_fvm, step_y, (step_x, step_y),
"FVM")
    plot_temp_profile_x(temperature_fdm, step_x, (step_x, step_y),
"FDM")
    plot_temp_profile_y(temperature_fdm, step_y, (step_x, step_y),
"FDM")

    step_x, step_y = 0.5, 0.5
    cells_x, cells_y = calculate_grid_cells(domain_width,
domain_height, step_x, step_y)
    temperature_fvm, coeff_matrix, load_vector =
solve_heat_fvm(cells_x, cells_y, thermal_conductivity, heat_flux,
step_x, step_y)
    temperature_fdm = solve_heat_fdm(cells_x, cells_y,
thermal_conductivity, heat_flux, step_x, step_y)

    plot_heatmap(temperature_fvm, step_x, step_y, title="Temperature
Distribution "f'{step_x}x{step_y}'" resolution (FVM)", cmap="plasma")
    plot_heatmap(temperature_fdm, step_x, step_y, title="Temperature
```

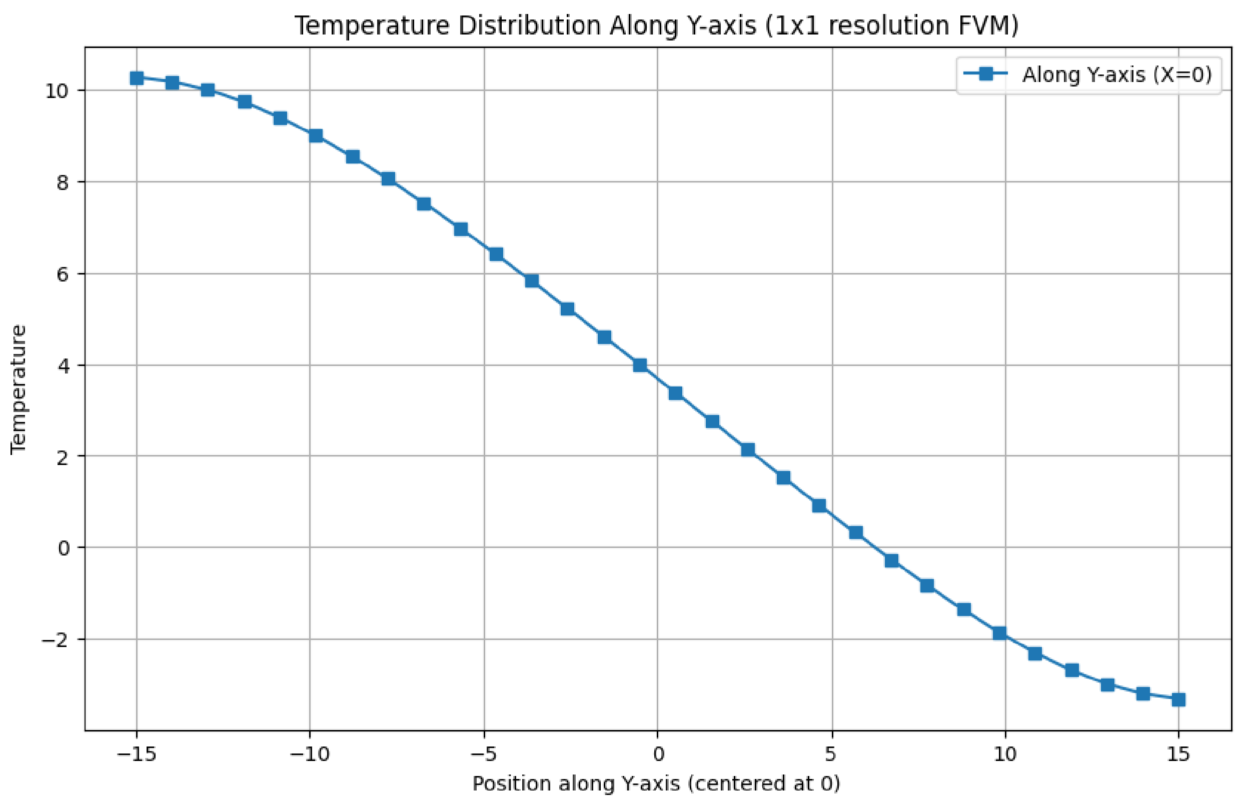
```
Distribution "f'{step_x}x{step_y}'" resolution (FDM)", cmap="inferno")

    plot_temp_profile_x(temperature_fvm, step_x, (step_x, step_y),
"FVM")
    plot_temp_profile_y(temperature_fvm, step_y, (step_x, step_y),
"FVM")
    plot_temp_profile_x(temperature_fdm, step_x, (step_x, step_y),
"FDM")
    plot_temp_profile_y(temperature_fdm, step_y, (step_x, step_y),
"FDM")

if __name__ == "__main__":
    main()
```

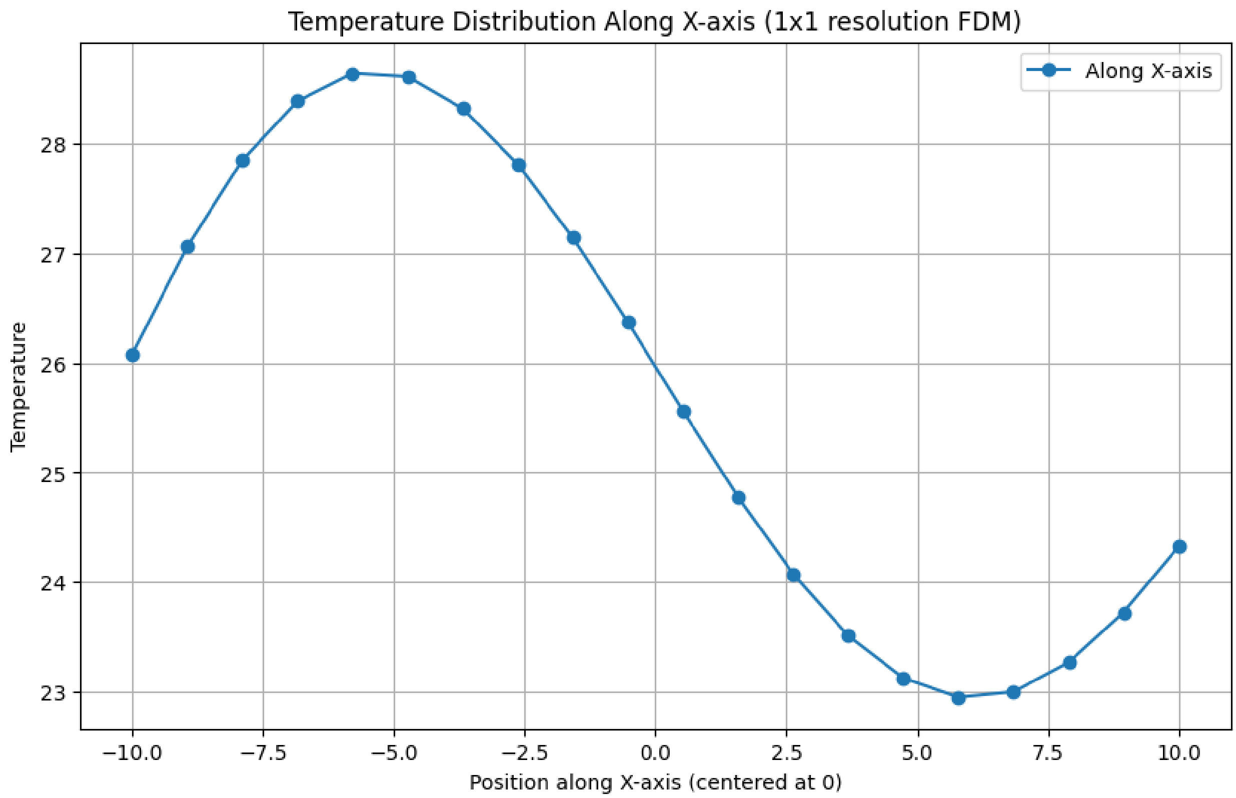


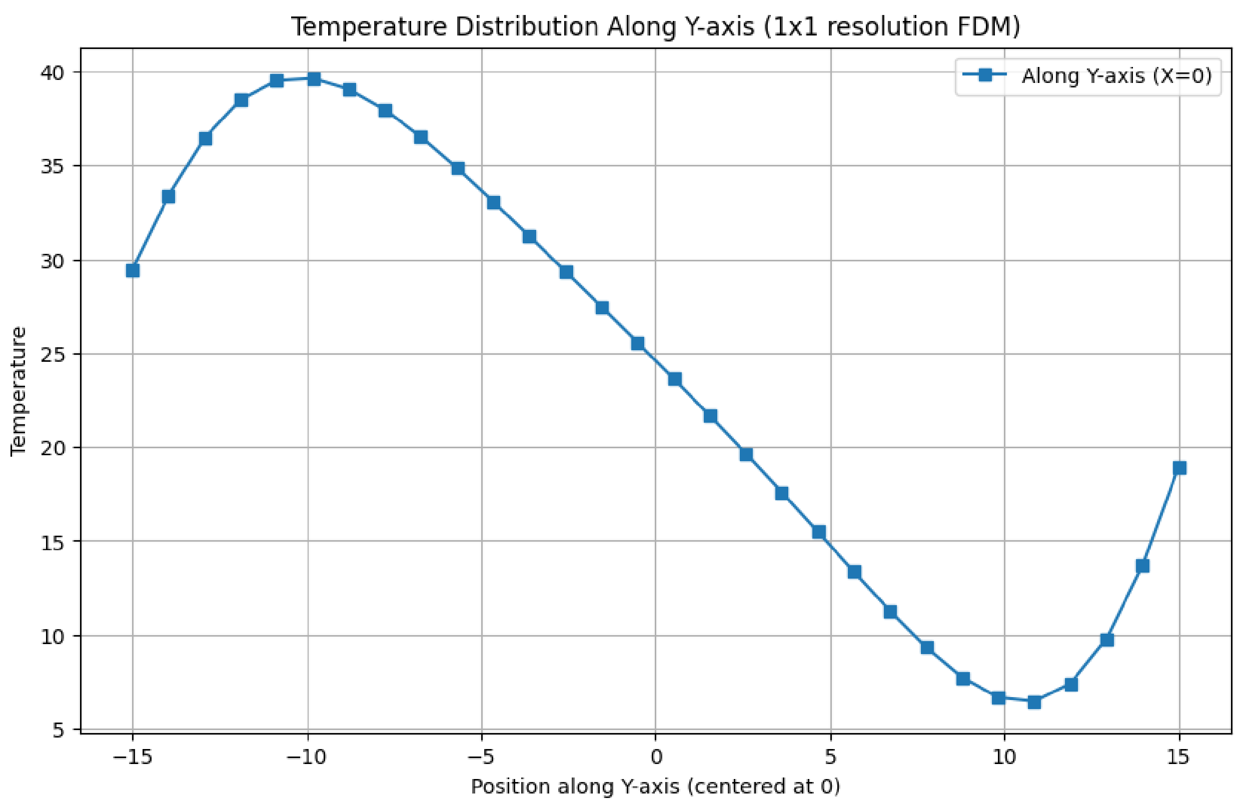


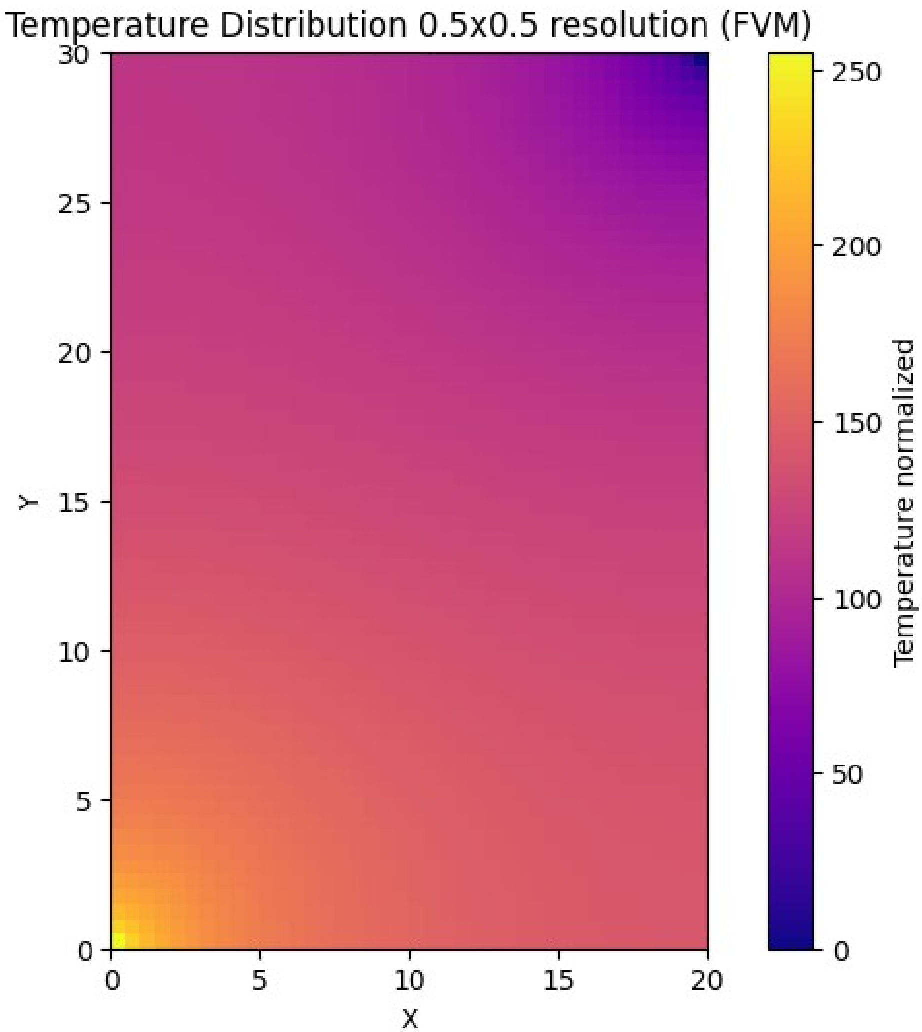
Temperature Distribution 1x1 resolution (FDM)
Y
X
0
5
10
15
20
25
30
0
5
10
15
20
Temperature normalized
0
50
100
150
200
250

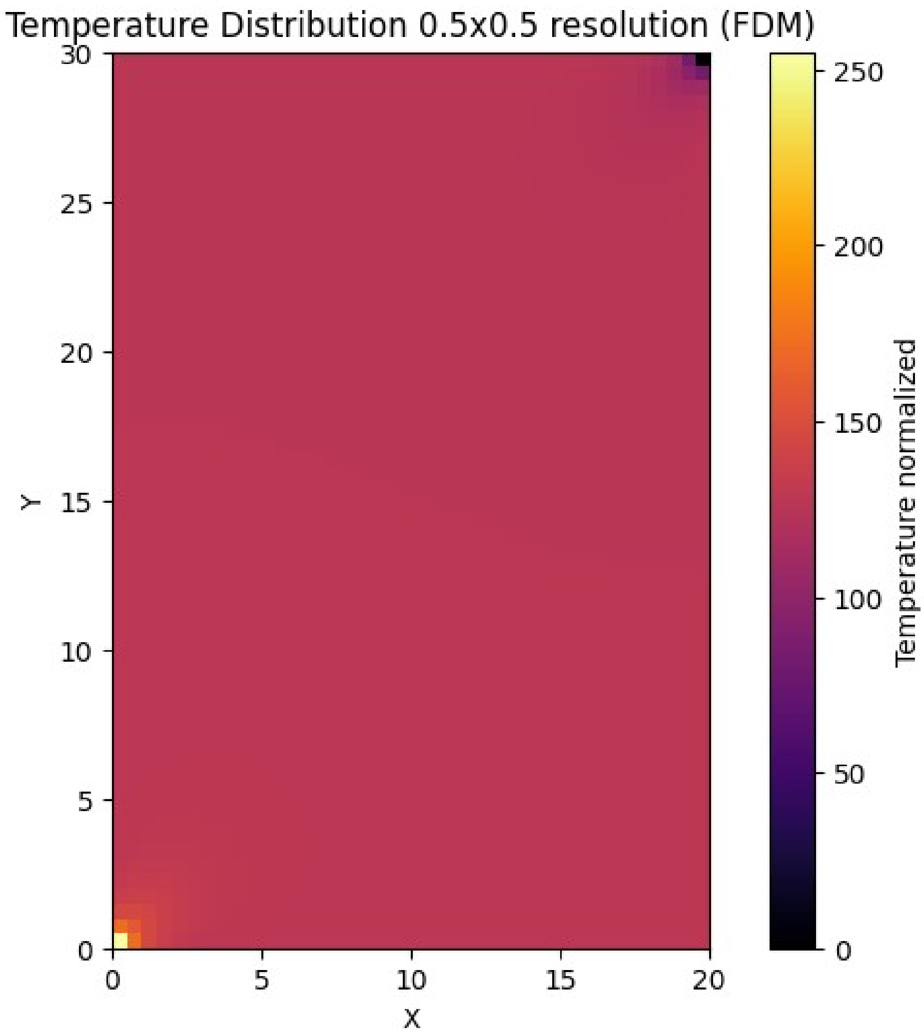

Temperature Distribution Along X-axis (1x1 resolution FVM)
Along X-axis
Temperature
5.5
5.0
4.5
4.0
3.5
3.0
2.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Position along X-axis (centered at 0)


Temperature Distribution Along Y-axis (1x1 resolution FVM)
Along Y-axis (X=0)
Temperature
10
8
6
4
2
0
−2
−15
−10
−5
0
5
10
15
Position along Y-axis (centered at 0)

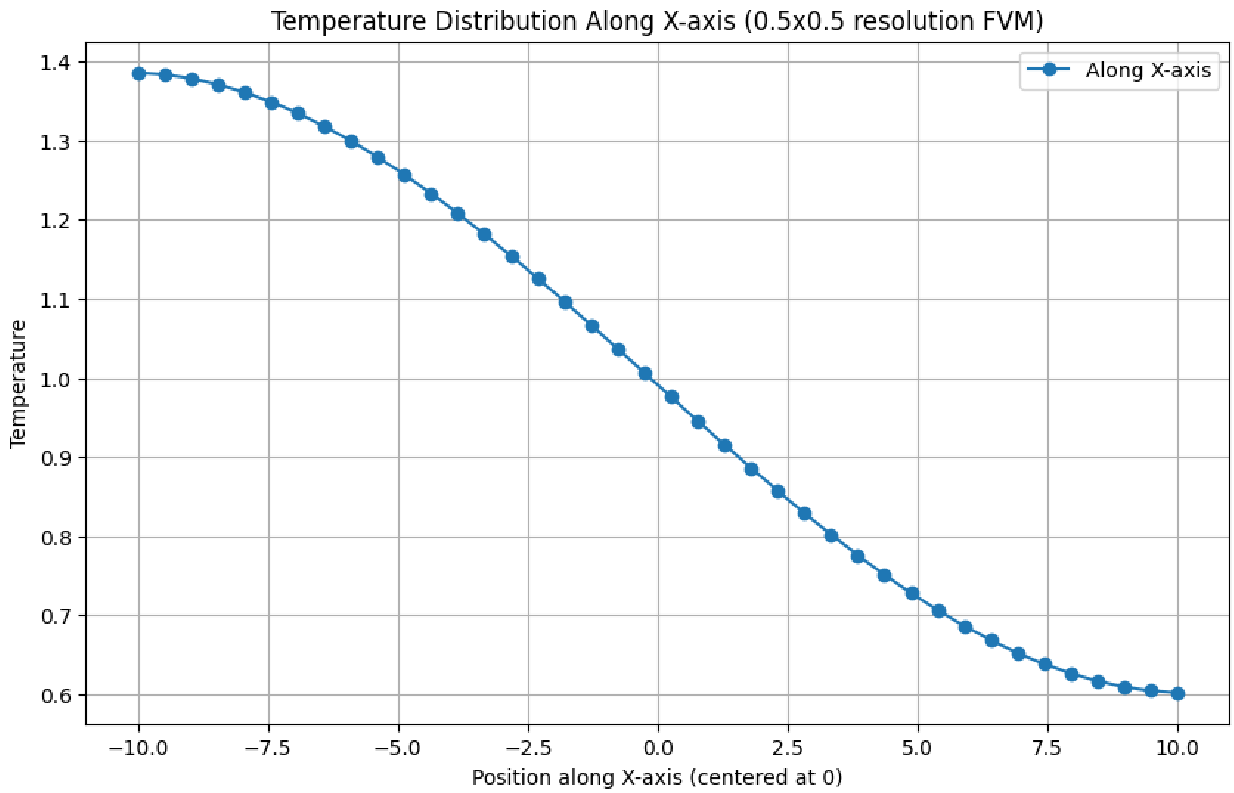

Temperature Distribution Along X-axis (1x1 resolution FDM)
Along X-axis
Temperature
28
27
26
25
24
23
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Position along X-axis (centered at 0)

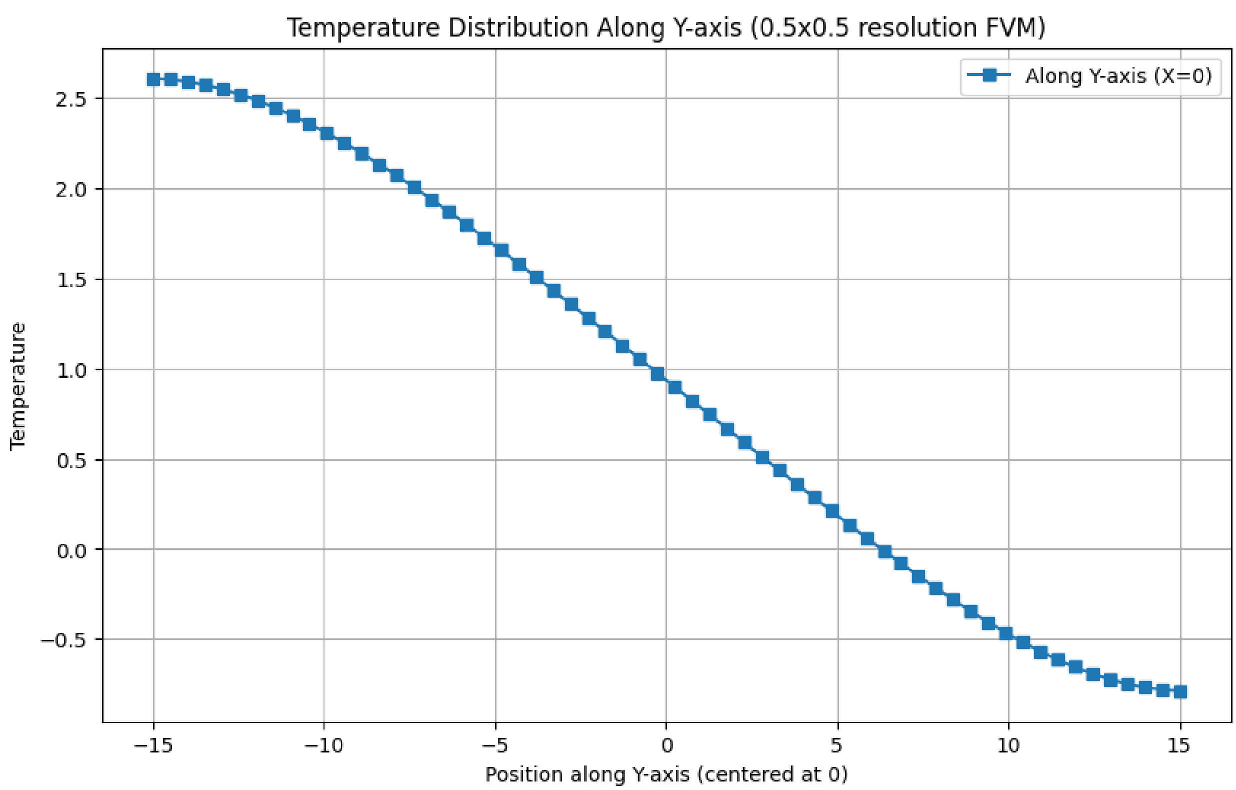

Temperature Distribution Along Y-axis (1x1 resolution FDM)
Along Y-axis (X=0)
Temperature
40
35
30
25
20
15
10
5
−15
−10
−5
0
5
10
15
Position along Y-axis (centered at 0)

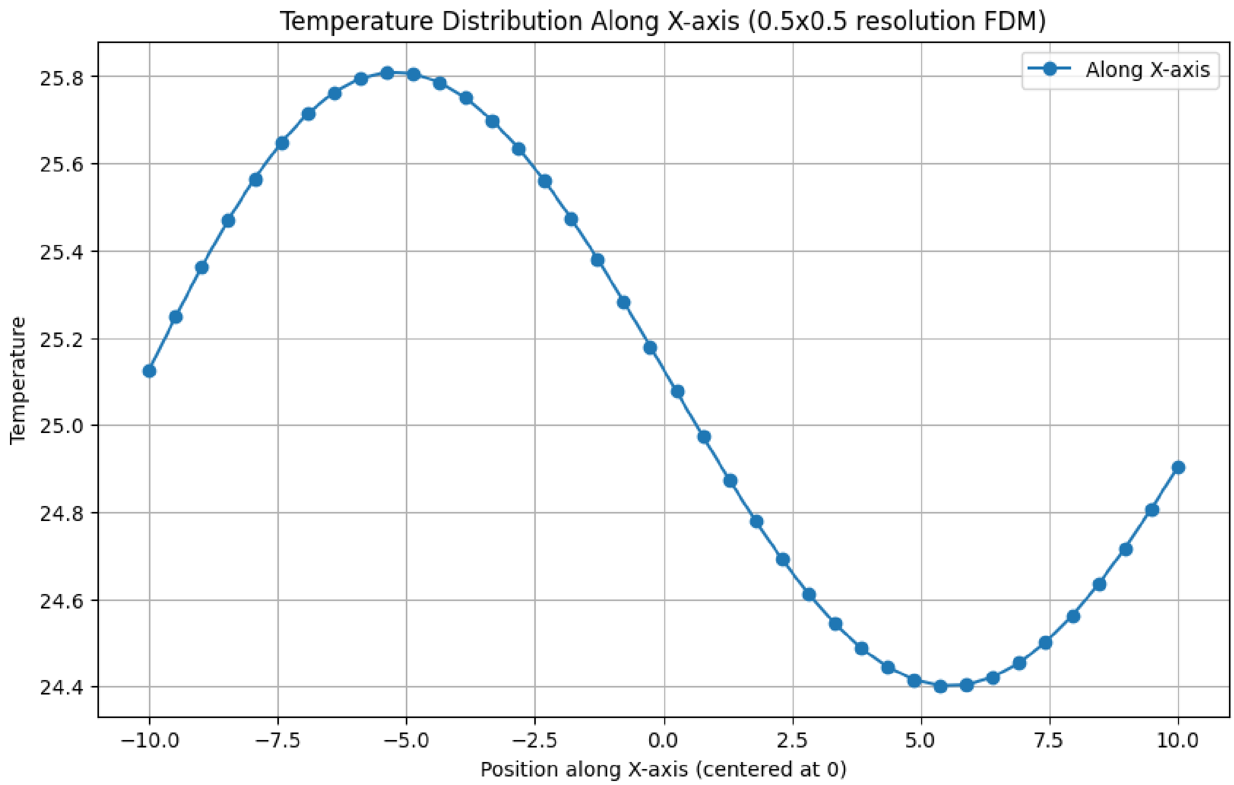

Temperature Distribution 0.5x0.5 resolution (FVM)
Y
X
Temperature normalized
0
5
10
15
20
25
30
0
5
10
15
20
0
50
100
150
200
250

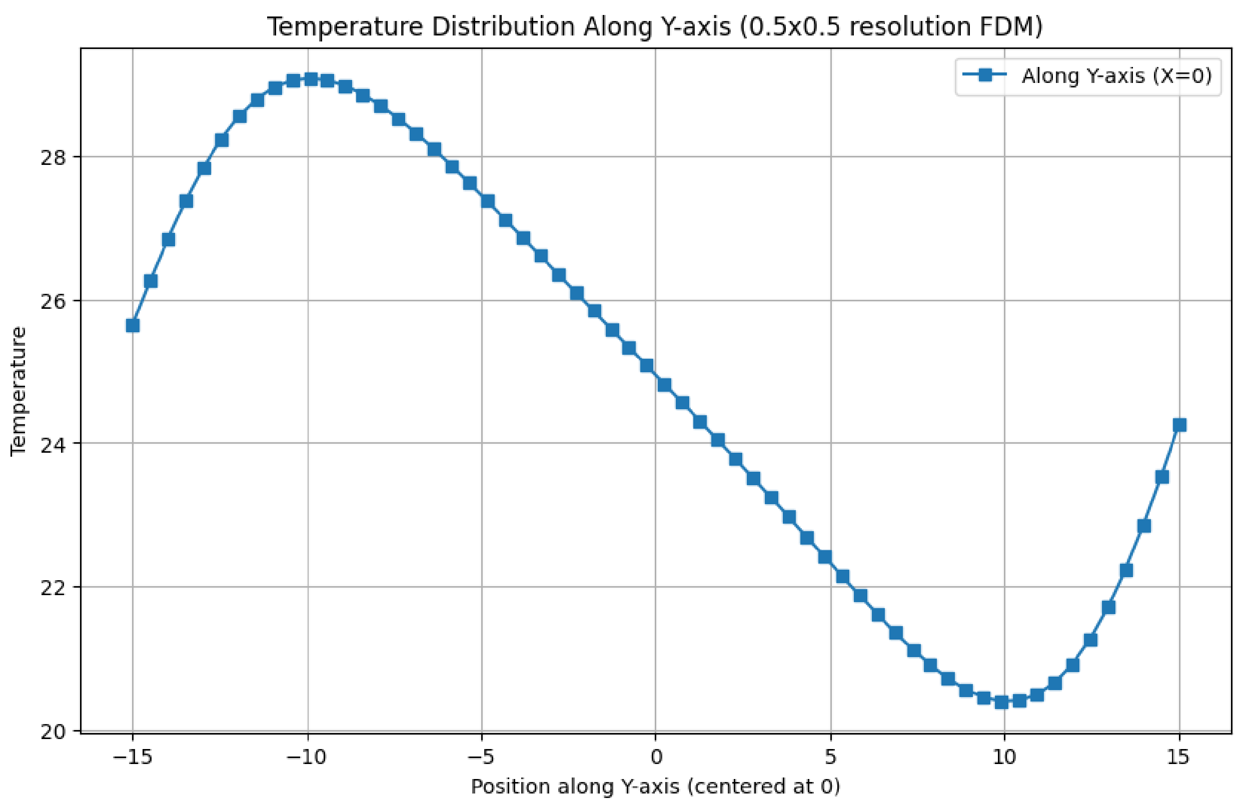


Temperature Distribution 0.5x0.5 resolution (FDM)
Y
X
Temperature normalized
0
5
10
15
20
25
30
0
50
100
150
200
250


Temperature Distribution Along X-axis (0.5x0.5 resolution FVM)
Along X-axis
Temperature
Position along X-axis (centered at 0)
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Temperature Distribution Along Y-axis (0.5x0.5 resolution FVM)
Along Y-axis (X=0)
Temperature
Position along Y-axis (centered at 0)
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−15
−10
−5
0
5
10
15


Temperature Distribution Along X-axis (0.5x0.5 resolution FDM)
Along X-axis
Temperature
Position along X-axis (centered at 0)
25.8
25.6
25.4
25.2
25.0
24.8
24.6
24.4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Temperature Distribution Along Y-axis (0.5x0.5 resolution FDM)
Along Y-axis (X=0)
Temperature
Position along Y-axis (centered at 0)
28
26
24
22
20
−15
−10
−5
0
5
10
15