ME5204 Finite Element Analysis

Assignment 4

Pratham Sunkad

ME21B145

Contents

1 Problem 1

1.1 Problem Statement

The heat source with a volumetric heat generation $Q_0(x)$, following a two-dimensional Gaussian distribution centered at the Gajendra circle induces heat into the IITM region. The mission is to safely evacuate people from the campus for which the temperature distribution is required. Hence, our aim is to study the temperature behavior in our computational domain over space and time.

The tasks are as follows:

- Identify the optimum mesh size required to analyze the temperature behavior by performing a mesh convergence test.
- Based on the CFL criteria, determine the optimum time step size by performing a time convergence test.
- Choose any three locations in the academic, residential, and hostel zones to compute the time taken to reach $46°C$ in each of them.
- Draw observations from the results obtained with implicit and explicit schemes for Neuman and Dirichlet boundary conditions.

1.2 Weak Form Formulation

The governing differential equation for the transient heat conduction is given by:

$$\rho C_p \frac{\partial \theta(x,t)}{\partial t} = \nabla \cdot (\kappa \nabla \theta(x,t)) + Q_0(x) \tag{1}$$

Invoking the Galerkin Orthogonality property, it can be stated that:

$$\int_\Omega R(x,t)v(x,t)\,d\Omega = 0 \quad \forall v(x,t) \tag{2}$$

The residual $R(x,t)$ derived from the governing equation is given by:

$$R(x,t) = \rho C_p \frac{\partial \theta}{\partial t} - \nabla \cdot (\kappa \nabla \theta) - Q_0(x) \tag{3}$$

Given a vector F and a scalar v, the following result holds:

$$\nabla \cdot (vF) = (\nabla v) \cdot F + v(\nabla \cdot F) \tag{4}$$

From Gauss's divergence theorem for 2D surfaces:

$$\int_\Gamma F \cdot n\,ds = \int_\Omega \nabla \cdot F\,dA \tag{5}$$

Using these equations, the weak form can be obtained as:

$$\int_\Omega \rho C_p \frac{\partial \theta}{\partial t} v\,d\Omega + \int_\Omega \kappa \nabla \theta \cdot \nabla v\,d\Omega = \int_\Omega Q_0 v\,d\Omega + \int_\Gamma \kappa \frac{\partial \theta}{\partial n} v\,d\Gamma \tag{6}$$

This has to hold true for all $v(x,t)$, and hence we represent $v(x,t)$ as a linear combination of Lagrange basis functions. Similarly, $\theta(x,t)$ can also be represented as a linear combination of Lagrange basis functions. Our objective is to find $c_I(t)$.

$$v(x,t) = \sum_J \Phi_J v_J(t) \tag{7}$$

$$\theta(x,t) = \sum_I \Phi_I c_I(t) \tag{8}$$

v can be either of the same degree as $\theta(x,t)$ or different. We choose it to be of the same

degree (referred to as Bubnov-Galerkin) as it enables the formulation of a square matrix, providing a unique solution for $c_I(t)$. It is important to note that the coefficients c_I are time-dependent while the Φ_I are time-independent.

For time marching, we adopt a finite difference formulation. Within finite difference, we have a choice between explicit and implicit schemes.

1.2.1 Explicit Scheme

Substituting the explicit finite difference formulation:

$$\int_\Omega \rho C_p \frac{\theta^{n+1} - \theta^n}{\Delta t} v \, d\Omega + \int_\Omega \kappa \nabla \theta^n \cdot \nabla v \, d\Omega = \int_\Omega Q_0 v \, d\Omega + \int_\Gamma \kappa \frac{\partial \theta^n}{\partial n} v \, d\Gamma \tag{9}$$

θ^{n+1} is an unknown, bringing the known values to the RHS and rearranging gives us:

$$\underbrace{\int_\Omega \rho C_p \frac{\theta^{n+1}}{\Delta t} v \, d\Omega}_{\text{(mass term)}} = \underbrace{\int_\Omega Q_0 v \, d\Omega}_{\text{(linear term)}} + \underbrace{\int_\Gamma \kappa \frac{\partial \theta^n}{\partial n} v \, d\Gamma}_{\text{(boundary term)}} - \underbrace{\int_\Omega \kappa \nabla \theta^n \cdot \nabla v \, d\Omega}_{\text{(bilinear term)}} + \underbrace{\int_\Omega \rho C_p \frac{\theta^n}{\Delta t} v \, d\Omega}_{\text{(mass term)}}$$

The mass term forms the mass matrix M and the bilinear term forms the stiffness matrix K. The simplified expression for the explicit scheme is as shown:

$$\frac{M}{\Delta t} C^{t+\Delta t} = \left(\frac{M}{\Delta t} - K \right) C^t + Q + BC \tag{10}$$

$C^{t+\Delta t}$ is the temperature coefficients vector at $t + \Delta t$, C^t is the temperature coefficients vector at t, Q is the source term vector, and BC is the boundary conditions.

1.2.2 Implicit Scheme

Substituting the implicit finite difference formulation:

$$\int_\Omega \rho C_p \frac{\theta^{n+1} - \theta^n}{\Delta t} v \, d\Omega + \int_\Omega \kappa \nabla \theta^{n+1} \cdot \nabla v \, d\Omega = \int_\Omega Q_0 v \, d\Omega + \int_\Gamma \kappa \frac{\partial \theta^{n+1}}{\partial n} v \, d\Gamma \tag{11}$$

θ^{n+1} is an unknown, bringing the known values to the RHS and rearranging gives us:

$$\underbrace{\int_\Omega \rho C_p \frac{\theta^{n+1}}{\Delta t} v \, d\Omega}_{\text{(mass term)}} + \underbrace{\int_\Omega \kappa \nabla \theta^{n+1} \cdot \nabla v \, d\Omega}_{\text{(bilinear term)}} = \underbrace{\int_\Omega Q_0 v \, d\Omega}_{\text{(linear term)}} + \underbrace{\int_\Gamma \kappa \frac{\partial \theta^{n+1}}{\partial n} v \, d\Gamma}_{\text{(boundary term)}} + \underbrace{\int_\Omega \rho C_p \frac{\theta^n}{\Delta t} v \, d\Omega}_{\text{(mass term)}}$$

The mass term forms the mass matrix M and the bilinear term forms the stiffness matrix K. The simplified expression for the implicit scheme is as shown:

$$\left(\frac{M}{\Delta t} + K \right) C^{t+\Delta t} = \frac{M}{\Delta t} C^t + Q + BC \tag{12}$$

The same notations as earlier are used.

$$M : \int_\Omega \begin{bmatrix} \phi_1\phi_1 & \phi_1\phi_2 & \cdots & \phi_1\phi_N \\ \phi_2\phi_1 & \phi_2\phi_2 & \cdots & \phi_2\phi_N \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N\phi_1 & \phi_N\phi_2 & \cdots & \phi_N\phi_N \end{bmatrix} d\Omega$$

$$K : \int_\Omega \kappa_x \begin{bmatrix} \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_1}{\partial x} & \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_2}{\partial x} & \cdots & \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_N}{\partial x} \\ \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_1}{\partial x} & \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_2}{\partial x} & \cdots & \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_N}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\phi_N}{\partial x}\frac{\partial\phi_1}{\partial x} & \frac{\partial\phi_N}{\partial x}\frac{\partial\phi_2}{\partial x} & \cdots & \frac{\partial\phi_N}{\partial x}\frac{\partial\phi_N}{\partial x} \end{bmatrix} + \kappa_y \begin{bmatrix} \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_1}{\partial y} & \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_2}{\partial y} & \cdots & \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_N}{\partial y} \\ \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_1}{\partial y} & \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_2}{\partial y} & \cdots & \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_N}{\partial y} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\phi_N}{\partial y}\frac{\partial\phi_1}{\partial y} & \frac{\partial\phi_N}{\partial y}\frac{\partial\phi_2}{\partial y} & \cdots & \frac{\partial\phi_N}{\partial y}\frac{\partial\phi_N}{\partial y} \end{bmatrix} d\Omega$$

$$C_t : \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}_t \quad \text{and} \quad C_{t+\Delta t} : \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}_{t+\Delta t}$$

$$Q : \quad \int_\Omega Q_0(x) \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix} d\Omega$$

The formulation of **M** and **Q** is the same as discussed in earlier assignments. The pseudocode for formulation of **K** is as follows:

```
def construct_bilinear(total_nodes, node_tags, triangle):
    n = total_nodes
    A_x = zeros((n,n))
    A_y = zeros((n,n))
    for i in range(len(node_tags)):
        nodes = node_tags[i]
        x1, y1, x2, y2, x3, y3 = triangle[i]
        jac = triangle_jacobian(x1, y1, x2, y2, x3, y3)
        pts = [[1/6, 2/3], [1/6, 1/6], [2/3, 1/6]]
        weights = [1/6, 1/6, 1/6]
        a_x = zeros((3,3))
        a_y = zeros((3,3))
        jacobian = jac_matrix(x1, y1, x2, y2, x3, y3)
        dphi = inv(jacobian) @ array([[-1,1,0],[-1,0,1]])
        for j in range(len(weights)):
            x = x1*(1-pts[j][0] - pts[j][1]) + x2*pts[j][0] + x3*pts[j][1]
            y = y1*(1-pts[j][0] - pts[j][1]) + y2*pts[j][0] + y3*pts[j][1]
```

```
            kappa = properties(x, y)[0]
            N_x = dphi[0,:]
            N_y = dphi[1,:]
            N_x = N_x.reshape((3, 1))
            N_y = N_y.reshape((3, 1))
            a_x += (N_x @ N_x.T) * jac * weights[j] * kappa
            a_y += (N_y @ N_y.T) * jac * weights[j] * kappa
        for k in range(3):
            for l in range(3):
                A_x[int(nodes[k]-1), int(nodes[l]-1)] += a_x[k][l]
                A_y[int(nodes[k]-1), int(nodes[l]-1)] += a_y[k][l]

    K = A_x + A_y
    return K
```

1.3 Region-wise dependence of κ, ρ and C_p

The IITM campus can be segregated into three major regions:- forest, water - bodies and buildings. Each of these regions will possess different values of κ, ρ and C_p. Hence, we will have to segment the map and assign the corresponding property values. Using the interactive ginput interface of matplotlib, mark points on the boundary of the above regions to distinguish them. Using the `polygon.contains_point` , we can check if a point lies inside or outside the path formed by connecting the boundary points.

```
def properties(x,y):
    if (check_point_in_region(x, y, 'instilake') == True or
        check_point_in_region(x, y, 'pond1') == True or
        check_point_in_region(x, y, 'pond2') == True):
        return 0.6 * 5 * 1e8, 1000 * (scale ** 3), 4184

    if check_point_in_region(x, y, 'forest') == True:
        return 0.128*5*1e8, 450*(scale**3), 1530
```

```
    else:
        return 0.1152*5*1e8, 405*(scale**3), 1377
```

The κ values of the regions are scaled up by a factor of 5×10^8. This is due to two main reasons: one being that we are given only κ of a single tree, but for computing the effective κ of the tree region, we must account for all the trees, which will have a higher κ value. Furthermore, having small κ will result in exorbitantly high temperatures (order of 10^{11}) at GC and its vicinity, which is physically impractical. To obtain feasible temperatures in the computational domain, we will have to assume large values for κ.

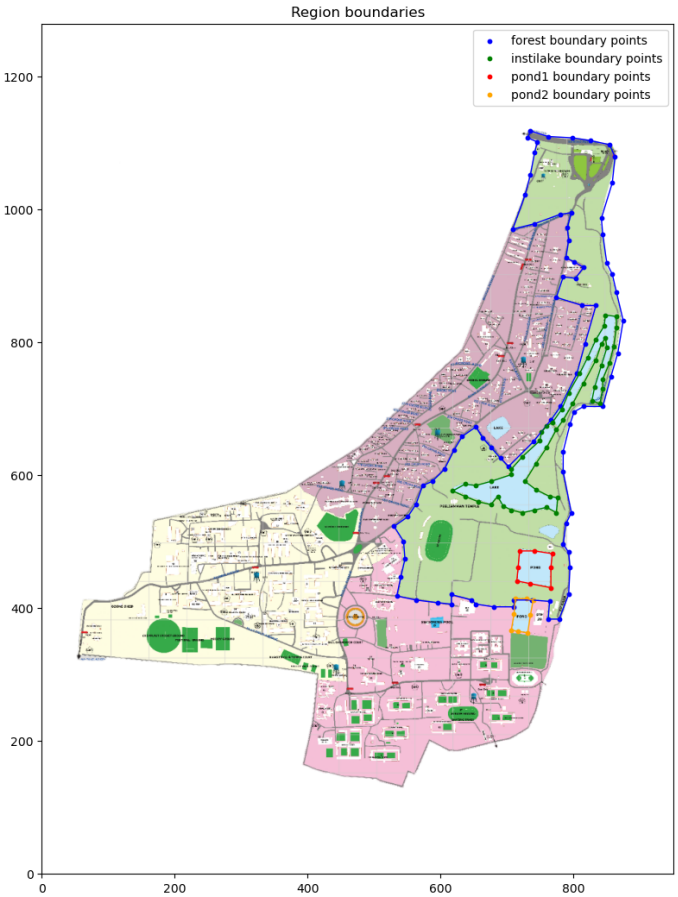


Figure 1: Different Regions in the IITM Map

1.4 Mesh Convergence

In order to choose the optimum mesh size, a convergence test must be performed on meshes different number of elements. The time step size is also undetermined; hence, we solve the steady-state problem. There is no temporal term for the steady-state formulation, hence no mass term:

$$KC_{\text{steady}} = Q + BC$$

There are two variants for the boundary conditions (BC), Neuman and Dirichlet. In Neuman, assuming the boundaries to be adiabatic, we obtain

$$\frac{\partial \theta}{\partial n} = 0,$$

which makes the BC a zero vector. It is known that K is a singular matrix. Hence, we cannot obtain a unique solution for steady-state temperatures when the Neuman boundary condition is imposed.

In Dirichlet, the boundary nodes have a fixed temperature. Hence, some of the coefficients in the C_{steady} vector are already known. We delete the rows and columns corresponding to boundary coefficients from the K matrix to solve only for the remaining coefficients. Accordingly, we modify the terms in the Q and BC vectors. The sub-matrix of K is invertible. Hence, we can obtain the unique solution for the remaining coefficients. The boundary nodes are fixed at an ambient temperature of $36°C$. The code is as shown below:

```
def static_solution(BC, total_nodes, boundary_nodes, A_x, A_y, f):
    t_amb = 309
    K = A_x + A_y
    boundary = boundary_nodes - 1
```

```
n = total_nodes
C = np.zeros((n,1))
C[boundary] = t_amb
unknown_indices = np.setdiff1d(np.arange(n), boundary)
# Extract submatrix K_uu (rows and columns corresponding to unknowns)
K_uu = K[np.ix_(unknown_indices, unknown_indices)]
K_uu_2 = K[np.ix_(unknown_indices, boundary)]
rhs = f[unknown_indices] - (K_uu_2 @ C[boundary])
C_uu = np.linalg.inv(K_uu) @ rhs
# Place the solved values back into the original C array \
# at the correct indices
C[unknown_indices] = C_uu
return C
```

A random point in the computational domain is picked, and its steady-state temperatures are measured across different mesh sizes. As the mesh gets finer, the temperatures should converge to a value.

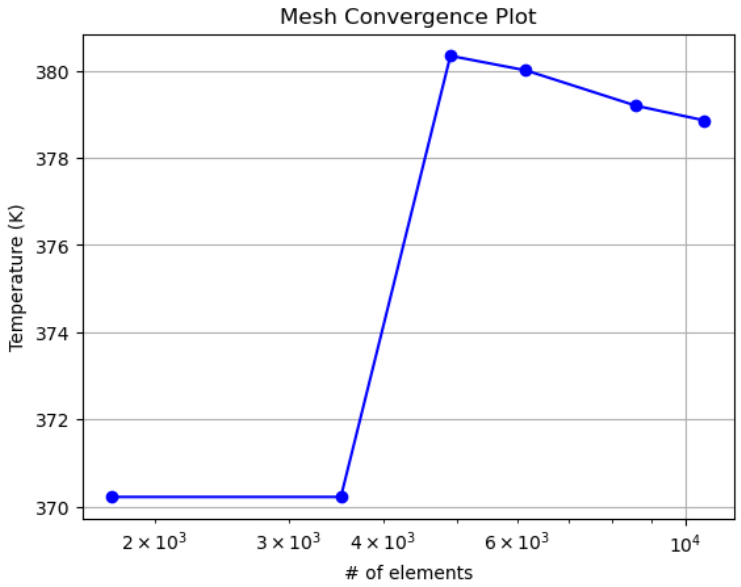


Figure 2: Mesh Convergence Plot

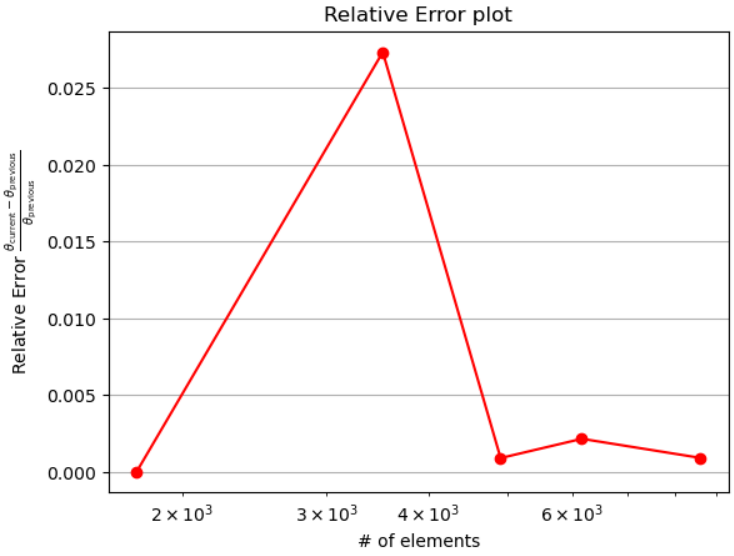
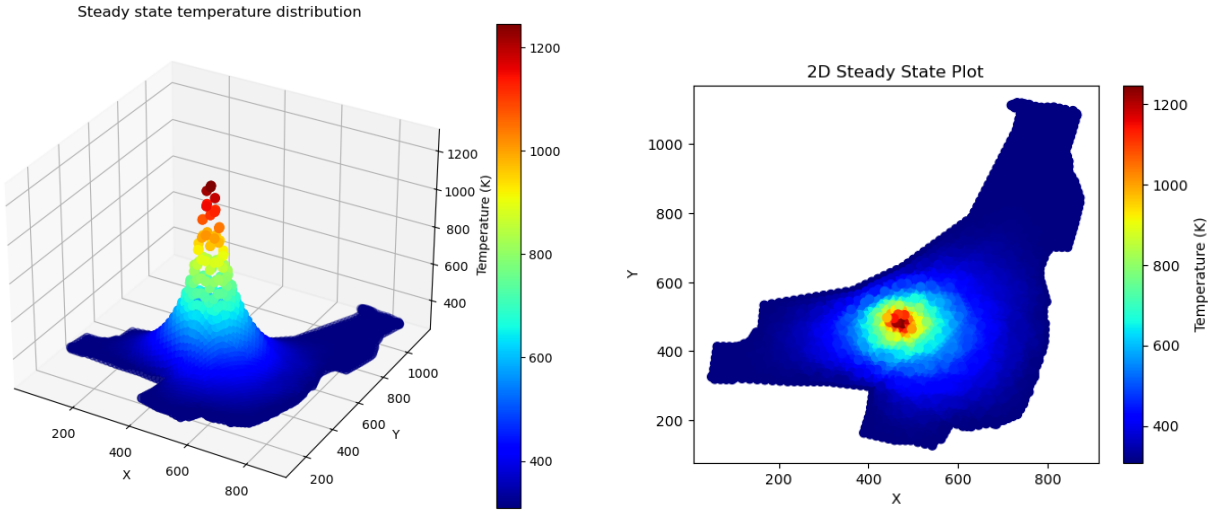


Figure 3: Relative Error Plot

For future proceedings, we use a mesh with Total elements = 8602.



(a) Steady State 3D Plot

(b) Steady State 2D Plot

Figure 4: Comparison of steady - state contour and 3D Plots

1.5 Time Convergence

As the mesh size is fixed now, we will have to choose the optimum time step size. This is analysis was done only for Dirichlet BC, but Neuman BC gives the exact same result. Explicit Scheme is conditionally stable. The CFL criteria for explicit scheme states:

$$\Delta t \leq 0.5 \cdot \frac{\rho C_p (\Delta x)^2}{\kappa}$$

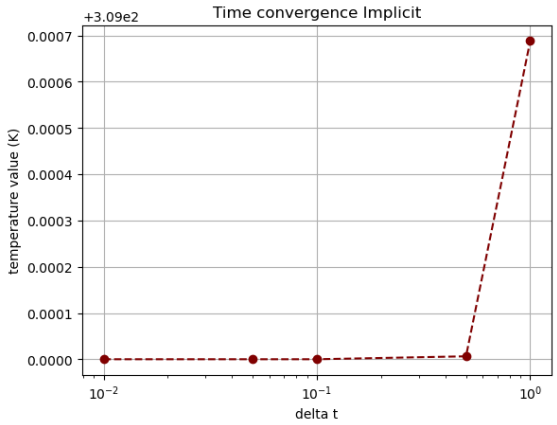


Figure 5: Time Convergence Plot for Implicit Scheme

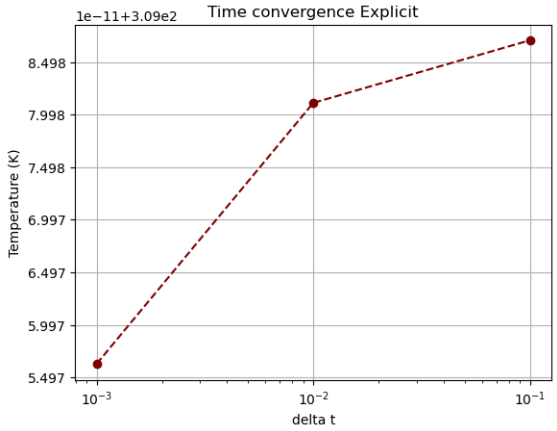


Figure 6: Time Convergence Plot for Explicit Scheme

For time convergence, choose an arbitrary point and time to check if the temperature values at that point and time are converging for different time steps.

Based on the above convergence study, Δt can be chosen to be 0.1 sec for the explicit scheme and 1 sec for the implicit scheme. Pick three points, one each in academic, hostel, and residential zone, and compute the time taken to reach 46$\circ$C. Locations closer to Gajendra Circle should be evacuated first as the temperature rise will be witnessed first. Hence, we sample three points in the vicinity of GC, as shown below, to find the worst-case scenario and compute the time taken using implicit, explicit, and Neuman and Dirichlet conditions.

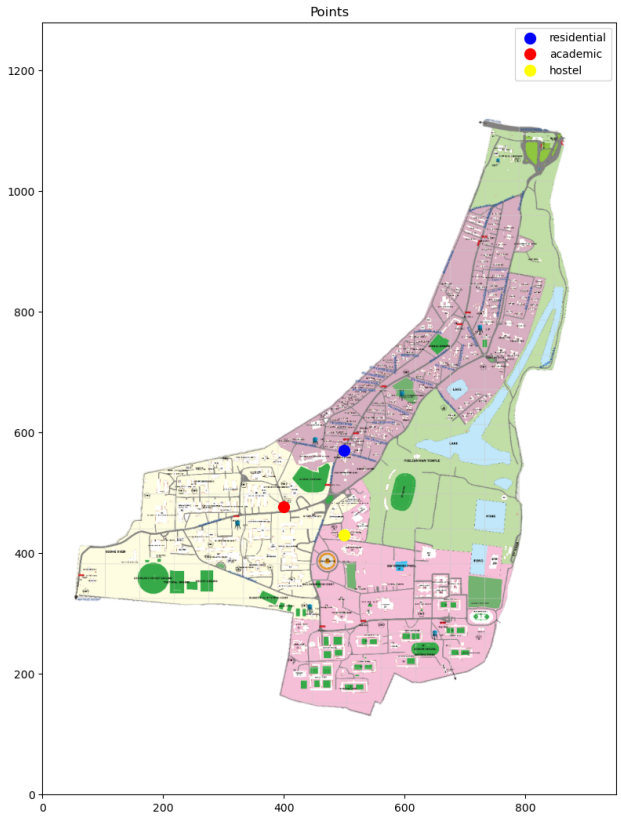
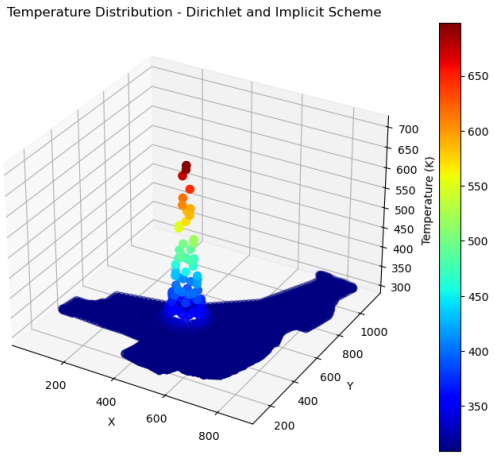


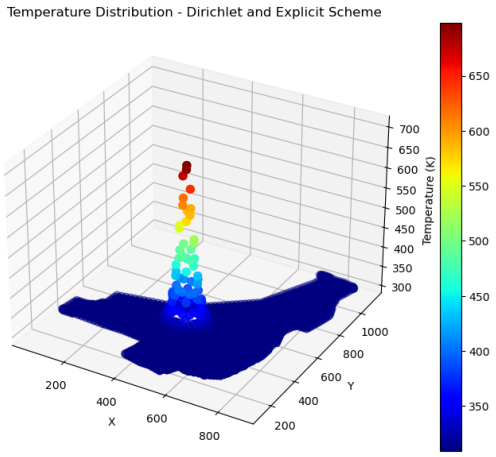
Figure 7: Sampled points in the Residential, Academic and Hostel Zone

Scheme	Boundary Condition	Academic Zone	Hostel Zone	Residential Zone
Implicit	Dirichlet	84 sec	89 sec	162 sec
Explicit	Dirichlet	83.9 sec	88.9 sec	161.5 sec
Implicit	Neuman	84 sec	89 sec	162 sec
Explicit	Neuman	83.9 sec	88.9 sec	161.5 sec

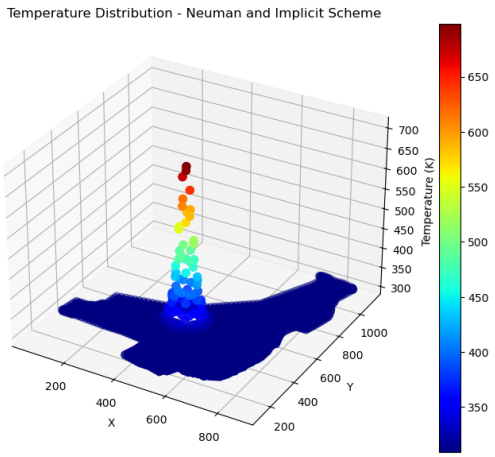
Table 1: Comparison of Schemes and Boundary Conditions across Different Zones



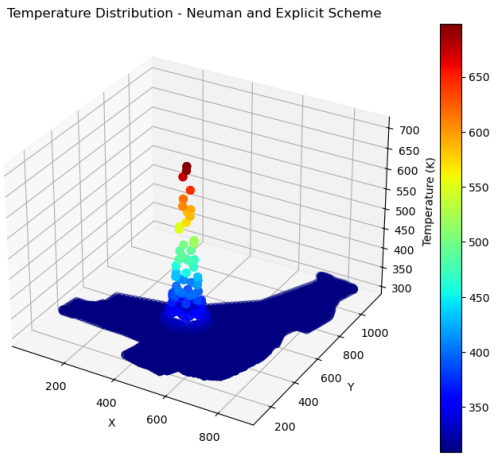
(a) Implicit Dirichlet



(b) Explicit Dirichlet



(c) Implicit Neuman



(d) Explicit Neuman

Figure 8: Temperature plots after reaching 46°C at the 3 points

Both explicit and implicit schemes yield the same temperature plots, which is a good sanity check for our results. Also, the plots are similar for both Neuman and Dirichlet boundary conditions. This is because, up till our time of interest, which is far before the steady state, both problem setups will behave similarly

Based on the results, the evacuation should proceed in the following order: the Academic zone must be evacuated within 84 seconds, followed by the hostel zone within 89 seconds, and finally, the residential zone within 162 seconds.

2 Problem 2

2.1 Problem Statement

A sound system is installed at the OAT, and the pressure distribution due to the sound needs to be studied. The governing equation for the acoustic pressure distribution is the Helmholtz equation, derived from the wave equation.

- The Forcing term includes $v = \frac{\partial p}{\partial n} = 0.1$ at the sound source.
- Perform a mesh convergence study to determine how far the computational domain should extend such that the pressure becomes sufficiently small.
- Measure the pressure at specified locations (main entrance, student gate, fountain, badminton court) and compare the computed values with actual measurements.
- Report and tabulate the comparison of pressure values (both measured and computed).

2.2 The Eigenvalue Problem

Given the wave equation:

$$\nabla^2 p' - \frac{1}{c^2}\frac{\partial^2 p'}{\partial t^2} = 0 \quad \text{in } \Omega$$

Substitute the harmonic solution $p' = pe^{i\omega t}$:

1. First, substitute $p' = pe^{i\omega t}$ into the given wave equation.

$$\nabla^2(pe^{i\omega t}) - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}(pe^{i\omega t}) = 0$$

2. Since p is a function of space only, and the time derivative only affects $e^{i\omega t}$, we get:

$$\nabla^2(pe^{i\omega t}) = e^{i\omega t}\nabla^2 p$$

$$\frac{\partial^2}{\partial t^2}(pe^{i\omega t}) = p\frac{\partial^2}{\partial t^2}(e^{i\omega t}) = p(-\omega^2 e^{i\omega t})$$

3. Substituting these results back into the original equation:

$$e^{i\omega t}\nabla^2 p + \frac{\omega^2}{c^2}pe^{i\omega t} = 0$$

4. Factor out $e^{i\omega t}$ from both terms:

$$e^{i\omega t}\left(\nabla^2 p + \frac{\omega^2}{c^2}p\right) = 0$$

Since $e^{i\omega t} \neq 0$, we are left with the Helmholtz equation:

$$\nabla^2 p + \frac{\omega^2}{c^2}p = 0$$

Eigenvalue problem: The Helmholtz equation can be treated as an eigenvalue problem

with $\lambda = \frac{\omega^2}{c^2}$, so we have:

$$\nabla^2 p + \lambda p = 0$$

2.3 Weak Form of the Helmholtz Equation

The strong form of the Helmholtz equation is:

$$\nabla^2 p + \lambda p = 0 \quad \text{in } \Omega$$

To derive the weak form, we multiply the equation by a test function v and integrate over the domain Ω:

$$\int_\Omega v \left(\nabla^2 p + \lambda p\right) \, d\Omega = 0$$

Using integration by parts for the Laplacian term:

$$-\int_\Omega \nabla v \cdot \nabla p \, d\Omega + \int_\Gamma v \frac{\partial p}{\partial n} \, d\Gamma + \lambda \int_\Omega v p \, d\Omega = 0$$

For a homogeneous Neuman boundary condition $\frac{\partial p}{\partial n} = 0$, the boundary integral vanishes, leaving:

$$\int_\Omega \nabla v \cdot \nabla p \, d\Omega + \lambda \int_\Omega v p \, d\Omega = 0$$

This is the weak form of the Helmholtz equation.

Matrix Form

In matrix notation, we approximate p and v using basis functions ϕ_i:

$$p \approx \sum_{i=1}^{N} p_i \phi_i, \quad v \approx \sum_{i=1}^{N} v_i \phi_i$$

Substituting into the weak form:

$$\sum_{i=1}^{N} v_i \left(\int_\Omega \nabla \phi_i \cdot \nabla \phi_j \, d\Omega \right) p_i + \lambda \sum_{i=1}^{N} v_i \left(\int_\Omega \phi_i \phi_j \, d\Omega \right) p_i = 0$$

This can be written as:

$$\mathbf{Kp} + \lambda \mathbf{Mp} = 0$$

where:

$$M : \int_\Omega \rho C_p \begin{bmatrix} \phi_1\phi_1 & \phi_1\phi_2 & \cdots & \phi_1\phi_N \\ \phi_2\phi_1 & \phi_2\phi_2 & \cdots & \phi_2\phi_N \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N\phi_1 & \phi_N\phi_2 & \cdots & \phi_N\phi_N \end{bmatrix} d\Omega$$

$$K : \int_\Omega \begin{bmatrix} \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_1}{\partial x} & \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_2}{\partial x} & \cdots & \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_N}{\partial x} \\ \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_1}{\partial x} & \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_2}{\partial x} & \cdots & \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_N}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\phi_N}{\partial x}\frac{\partial\phi_1}{\partial x} & \frac{\partial\phi_N}{\partial x}\frac{\partial\phi_2}{\partial x} & \cdots & \frac{\partial\phi_N}{\partial x}\frac{\partial\phi_N}{\partial x} \end{bmatrix} + \begin{bmatrix} \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_1}{\partial y} & \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_2}{\partial y} & \cdots & \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_N}{\partial y} \\ \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_1}{\partial y} & \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_2}{\partial y} & \cdots & \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_N}{\partial y} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\phi_N}{\partial y}\frac{\partial\phi_1}{\partial y} & \frac{\partial\phi_N}{\partial y}\frac{\partial\phi_2}{\partial y} & \cdots & \frac{\partial\phi_N}{\partial y}\frac{\partial\phi_N}{\partial y} \end{bmatrix} d\Omega$$

2.4 Mesh Convergence

In order to choose the optimum mesh size, a convergence test must be performed on meshes different number of elements. The Eigenvalue problem is solved without the forcing term.

The original equation is:

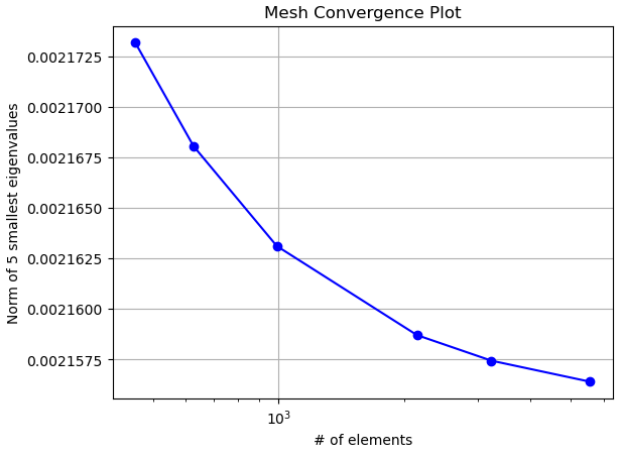
$$Kp + \lambda Mp = 0$$

Thus, the eigenvalue problem can be written as:

$$M^{-1}Kp = \lambda p$$

Where: - p is the eigenvector of $M^{-1}K$. - λ is the corresponding eigenvalue.

Eigenvalues of $M^{-1}K$ correspond to the λ. For mesh convergence study we take the norm of '5' such eigenvalues and plot it with the of elements.



(a) |eigenvalues| vs of elements

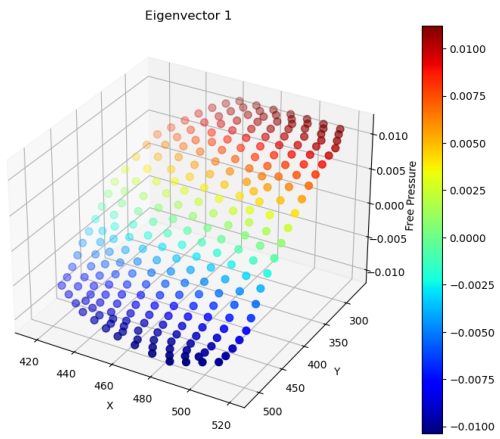
(b) Region meshed for analysis

Figure 9: Mesh Plots

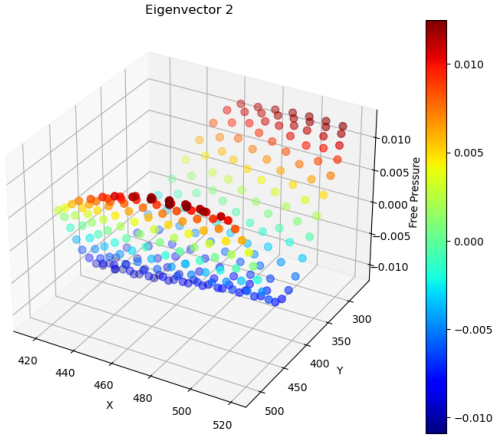
As we can see that the norm converges to a single value, yet a coarser mesh is doing as good as a fine mesh. Thus we can go ahead with coarser mesh where elements = ?

2.5 Eigenvectors

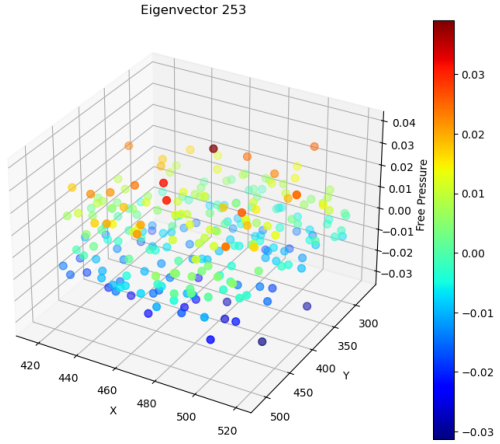
The eigenvectors calculated in mesh convergence study was the problem with no forcing term. The plots of these eigen vectors correspond to the free presssure and they vary significantly with the magnitude of the corresponding eigenvalue. Higher magnitude an the eigenvalue corresponds to a higher frequency solution.



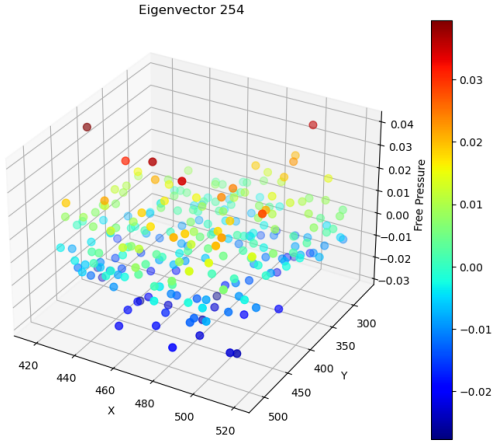
(a) Eigenvector 1



(b) Eigenvector 2



(c) Eigenvector 253



(d) Eigenvector 254

Figure 10: Plotting top 2 eigenvectors with the lowest and highest eigenvalues

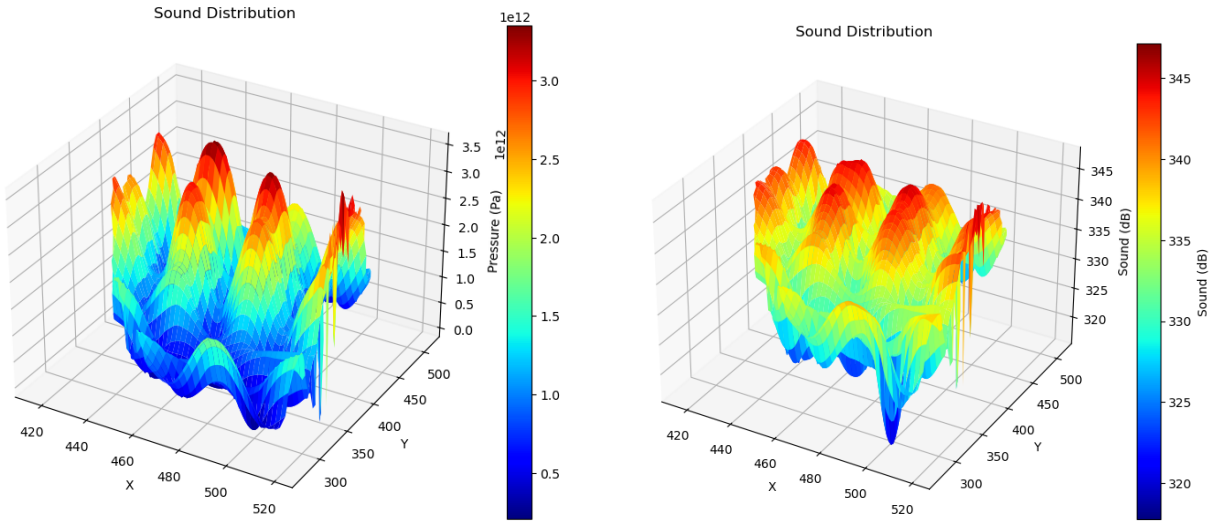
2.6 Pressure

Since they are so many eigenvalues which satisfy the helmholtz equation, we get many solutions to the given equation which now includes the forcing term. The forcing term is zero everywhere except at the source.

$$\mathbf{Kp} + \lambda\mathbf{Mp} = \mathbf{F}$$

The approach here is to take 5 eigenvalues and compute their corresponding pressure distributions, then take their RMS (root mean square) value.

The 5 eigenvalues chosen giving importance to the fact that there is a higher frequency solutions are more likely to occur



(a) Pressure Distribution (b) Decibel Level

Figure 11: 3D Plots for Sound Distribution

Location	Measured Value	Calculated Value
OAT Main Gate	63 dB	341.30 dB
OAT Students Gate	65 dB	340.99 dB
Fountain	54 dB	339.13 dB
Ball Badminton Court	49 dB	337.58 dB

Table 2: Comparison of Measured and Calculated dB Values at Different Locations

A possible reason why the calculated value is much higher than the measured value is that there is nothing obstructing the sound waves from propagating in our analysis. Whereas in real life we have buildings, trees and inanimate objects causing hindrance while propagating.

****