AE618A Assignment 3

Name: Vedant Joshi Roll No.: 180856

1)

1) Strong form of the Problem

Given, $g : \Gamma_g \to \mathbb{R}$, find, $u : \Omega \to \mathbb{R}$ such that

(Γ_g → inner and outer boundry)

$\nabla q = 0$ or $q_{i,i} = 0$ on Ω and

$u = g$ on Γ_g $g = 0$ for outer bc

$g = 1$ for inner bc

where, u → temperature field, q → heat flux.

Ω → domain, Γ_g → Dirchilet Boundry.

2)

2) Weak form of the problem

Let $\mathcal{S}$ be solution space and $\mathcal{V}$ be function space

$$\mathcal{S} = \{ u \mid u \in H^1, \; u = g \text{ on } \Gamma_g \}$$

$$\mathcal{V} = \{ \omega \mid \omega \in H^1, \; \omega = 0 \text{ on } \Gamma_g \}$$

given, $g : \Gamma_g \to \mathbb{R}$, find $u \in \mathcal{S}$ such that for all $\omega \in \mathcal{V}$

$$a(u, \omega) = 0$$

3)

3) Element level k and f using bilinear shape functions

$N_1 = \frac{1}{4}(1-\xi)(1-\eta)$ $N_3 = \frac{1}{4}(1+\xi)(1+\eta)$

$N_2 = \frac{1}{4}(1+\xi)(1-\eta)$ $N_4 = \frac{1}{4}(1-\xi)(1-\eta)$

$$k^e_{ab} = \int_{\Omega^e} (\nabla N_a)^T K (\nabla N_b) \, d\Omega$$

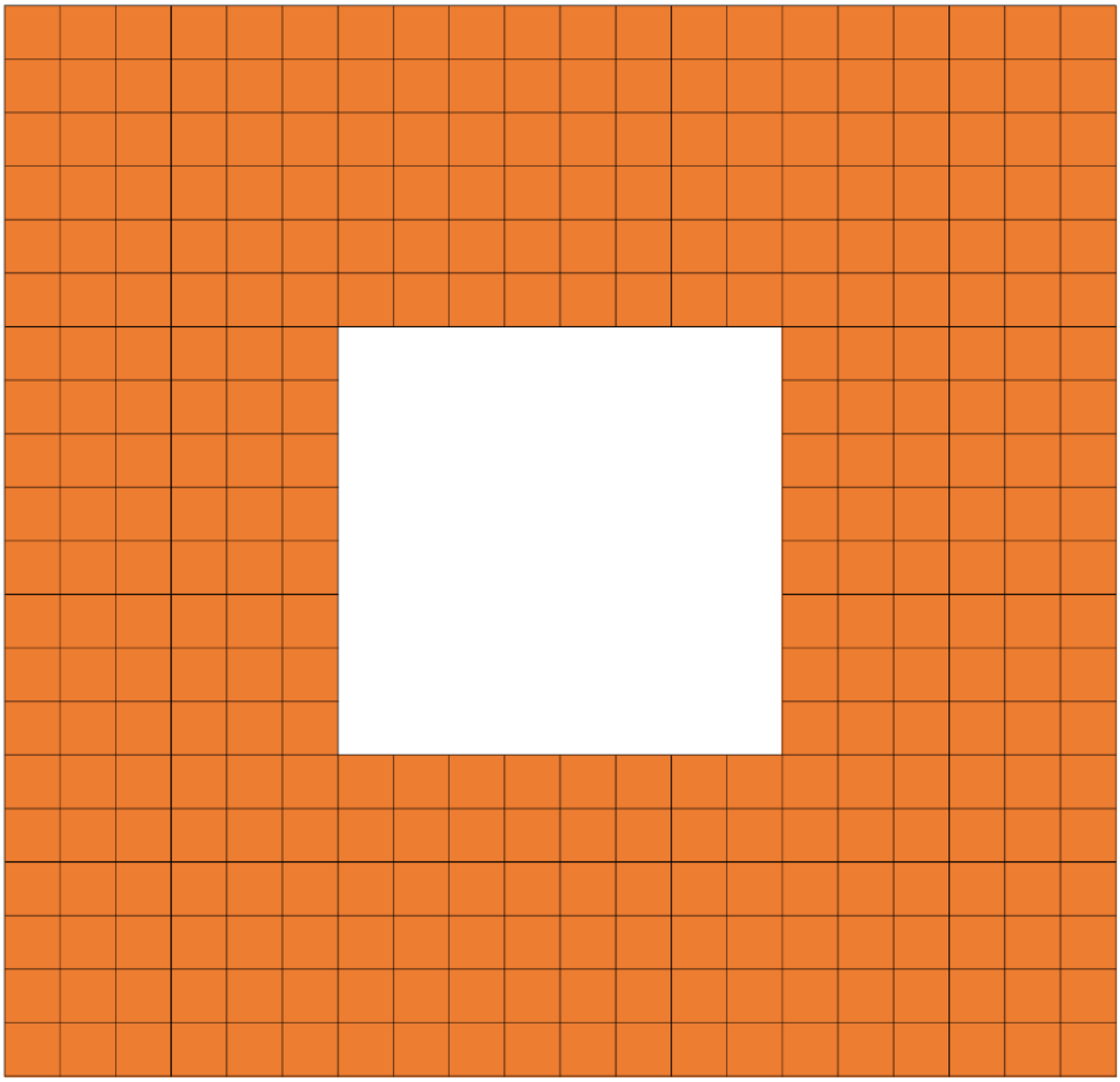
since $K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$= \int_{\Omega^e} \left(\frac{\partial N_a}{\partial x} \frac{\partial N_b}{\partial x} + \frac{\partial N_a}{\partial y} \frac{\partial N_b}{\partial y} \right) d\Omega$$

similarly $f^e_a = -\sum_{b=1}^{n_{en}} k^e_{ab} g^e_b$ (here, $n_{en} = 4$)

4) For this assignment I have choose 20mm square plate with 8mm square hole at centre.

Generated Mesh will have 392 Nodes, 336 Elements and 280 Equations.



```python
# Constructing IEN Matrix
IEN = [[ 0 for _ in range(336)] for _ in range(4)]

temp = 1
for e in range(120):
    if (temp%21)==0:
        temp += 1
    IEN[0][e] = temp
    IEN[1][e] = temp+1
    IEN[2][e] = temp+22
    IEN[3][e] = temp+21
    temp += 1

for e in range(120,216):
    if (e==126):
        temp += 7
    if ((e-120)%6)==0:
        temp += 1
    IEN[0][e] = temp
    IEN[1][e] = temp+1
    if e<126 or e>=210:
        IEN[2][e] = temp+22
        IEN[3][e] = temp+21
    else:
        IEN[2][e] = temp+15
        IEN[3][e] = temp+14
    temp += 1

for e in range(216, 336):
    if ((e-216)%20)==0:
        temp += 1
    IEN[0][e] = temp
    IEN[1][e] = temp+1
    IEN[2][e] = temp+22
    IEN[3][e] = temp+21
    temp += 1

# Constructing ID Array
ID = [-1 for _ in range(392)]

#list of nodes on boundry g1
B1 = list(range(1, 23)) + [42, 43, 63, 64, 84, 85, 105, 106, 126, 127, 133, 134, 135, 136, 13

for i in B1:
    ID[i-1] = 0

temp = 1
for i in range(392):
    if ID[i] != 0:
        ID[i] = temp
        temp += 1

# Constructing LM Matrix
LM = [[ 0 for _ in range(336)] for _ in range(4)]

for a in range(4):
    for e in range(336):
        LM[a][e] = ID[IEN[a][e]-1]
```

5) shape function subroutine

```
# Element level stiffness matrix
import math as mt

k = [[0 for _ in range(4)] for _ in range(4)]

N = [-1*mt.sqrt(3/5) , -1*mt.sqrt(3/5), -1*mt.sqrt(3/5), 0 , 0 , 0 , mt.sqrt(3/5) , mt.sqrt(3/5) , mt.sqrt(3/5)]
E = [-1*mt.sqrt(3/5) , 0 , mt.sqrt(3/5) , -1*mt.sqrt(3/5) , 0 , mt.sqrt(3/5) , -1*mt.sqrt(3/5) , 0 , mt.sqrt(3/5)]
weights = [ 25/81 , 40/81 , 25/81 , 40/81 , 64/81 , 40/81 , 25/81 , 40/81 ,  25/81]

def Ndx(a, y):
    if a == 0:
        return -0.5*(1-y)
    if a == 1:
        return 0.5*(1-y)
    if a == 2:
        return 0.5*(1+y)
    if a == 3:
        return -0.5*(1+y)

def Ndy(a, x):
    if a == 0:
        return -0.5*(1-x)
    if a == 1:
        return -0.5*(1+x)
    if a == 2:
        return 0.5*(1+x)
    if a == 3:
        return 0.5*(1-x)

for a in range(4):
    for b in range(4):
        k_val = 0
        for i in range(9):
            k_val += (Ndx(a, N[i]) * Ndx(b ,N[i]) + Ndy(a, E[i]) * Ndy(b, E[i]))*weights[i]
        k[a][b] = k_val

# printing the 4*4 element level stiffness matrix
print(k)
```

```
[[2.666666666666666, -0.6666666666666667, -1.3333333333333333, -0.6666666666666667], [-0.6666666666666667, 2.66666666
```

obtained value of element level stiffness matrix
[[2.666666666666666, -0.6666666666666667, -1.3333333333333333, -0.6666666666666667],
[-0.6666666666666667, 2.666666666666666, -0.6666666666666667, -1.3333333333333333],
[-1.3333333333333333, -0.6666666666666667, 2.6666666666666665, -0.6666666666666667],
[-0.6666666666666667, -1.3333333333333333, -0.6666666666666667, 2.666666666666666]]

6) program to assemble the global stiffness matrix and force vector

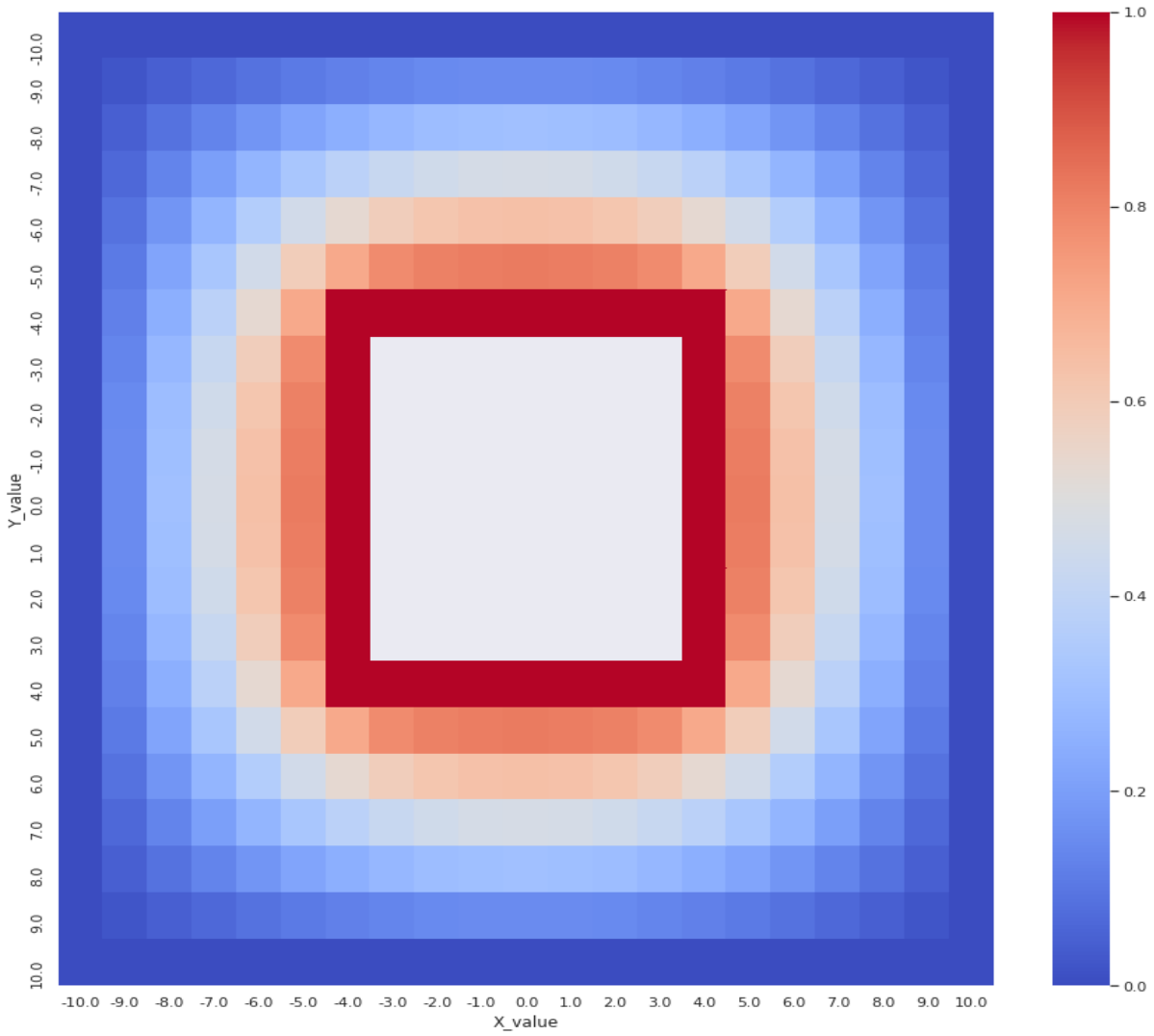
```
# Global stiffness matrix
K = [[0 for _ in range(280)] for _ in range(280)]

for i in range(336):
    for a in range(4):
        for b in range(4):
            p = LM[a][i]
            q = LM[b][i]
            if (p == 0) or (q == 0 ):
                continue
            else:
                K[p-1][q-1] = K[p-1][q-1] + k[a][b]

# Global Force Vector
#list of nodes on boundry g2
B2 = [ 106 , 107 , 108 , 109 , 110 , 111 , 112 , 113 , 114 , 115 , 126 , 127 , 13

F = [0 for _ in range(280)]

for i in B2:
    f = [0 for _ in range(4)]
    if i == 106 :
        for j in range(4):
            f[j] = - k[j][2]
    if i in [107 , 108 , 109 , 110 , 111 , 112 , 113 , 114]:
        for j in range(4):
            f[j] = -k[j][2]-k[j][3]
    if i == 115 :
        for j in range(4):
            f[j] = -k[j][3]
    if i in [126 , 138 , 150 , 162 , 174 , 186 , 198 , 210]:
        for j in range(4):
            f[j] = -k[j][1]-k[j][2]
    if i in [127 , 139 , 151 , 163 , 175 , 187 , 199 , 211]:
        for j in range(4):
            f[j] = -k[j][0]-k[j][3]
    if i == 222:
        for j in range(4):
            f[j] = -k[j][1]
    if i in [223 , 224 , 225 , 226 , 227 , 228 , 229 , 230]:
        for j in range(4):
            f[j] = -k[j][0]-k[j][1]
    if i == 231:
        for j in range(4):
            f[j] = -k[j][0]

    for a in range(4):
        p = LM[a][i-1]
        if p != 0:
            F[p-1] = F[p-1] + f[a]
```

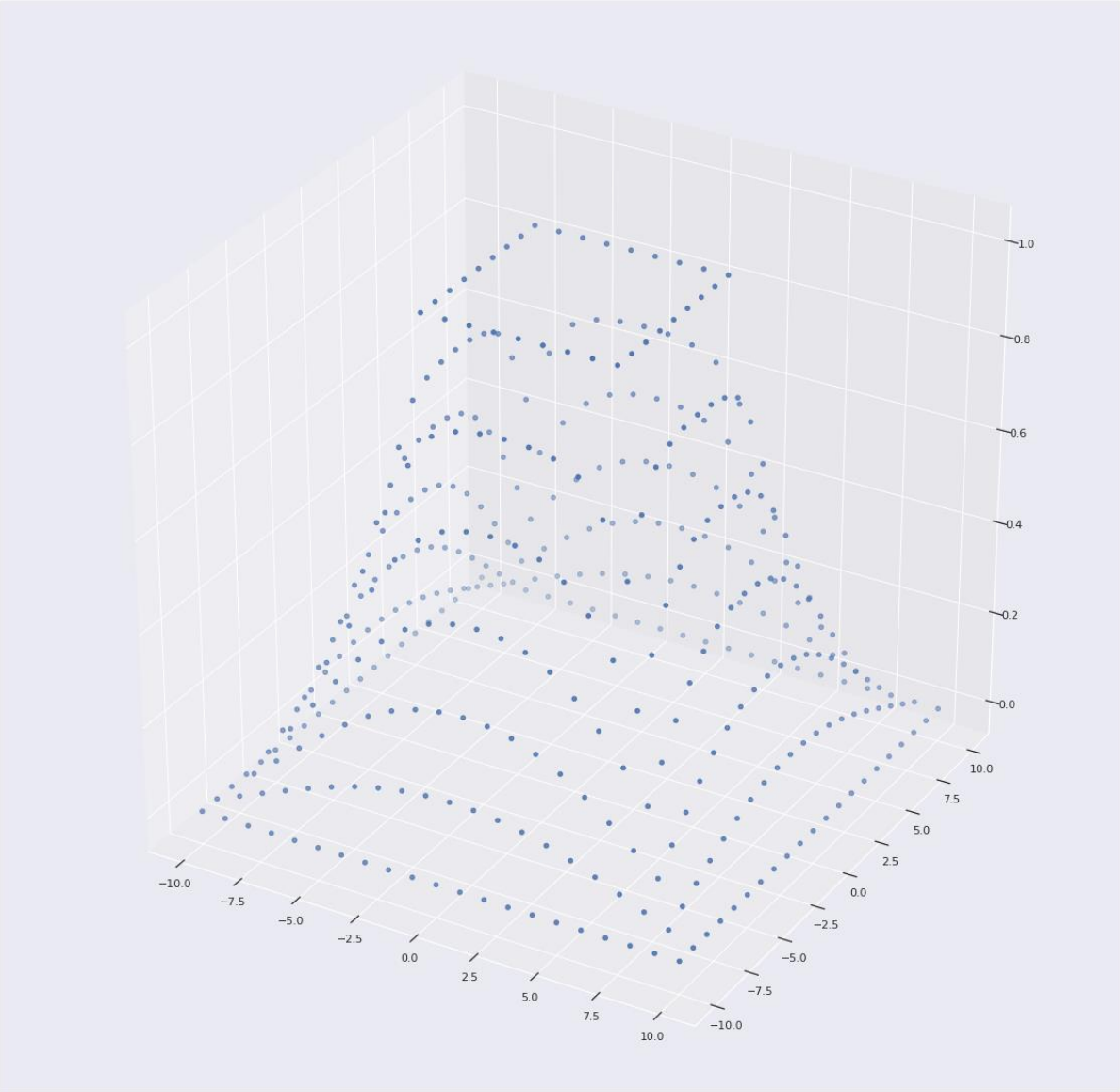
7) Find the solution

```
[4]  #Solving for the d vector
     import numpy as np

     K0 = np.matrix(K)
     F0 = np.array(F)

     kinv = np.linalg.inv(K0)

     d0 = np.matmul( kinv , F0 )

     X = [ 0 for _ in range(392)]
     Y = [ 0 for _ in range(392)]

     X0 = -10
     for x in range(147):
         if (X0 == 11) :
             X0 = -10
         X[x] = X0
         X0 += 1

     X0 = -10
     for x in range(147, 245):
         if (X0 == -3):
             X0 = 4
         if (X0 == 11):
             X0 = -10
         X[x] = X0
         X0 += 1

     X0 = -10
     for x in range(245, 392):
         if (X0 == 11):
             X0 = -10
         X[x] = X0
         X0 += 1

     Y0 = -11
     for y in range(147):
         if (y%21)==0:
             Y0 += 1
         Y[y] = Y0

     for y in range(147, 245):
         if (y-147)%14 == 0:
             Y0 += 1
         Y[y] = Y0

     for y in range(245, 392):
         if (y-245)%21 == 0:
             Y0 += 1
         Y[y] = Y0

     T = [ 0 for _ in range(392)]

     for i in range(392):
         eq = ID[i]
         if (eq != 0):
             T[i] = d0.__getitem__((0,eq-1))

     for i in B2:
         T[i-1] = 1
```

8) Plot: This is the obtained heatmap and scaatrplot3D

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns

x = np.array(X)
y = np.array(Y)
z = np.array(T)

df = pd.DataFrame.from_dict(np.array([x,y,z]).T)
df.columns = ['X_value','Y_value','Z_value']
df['Z_value'] = pd.to_numeric(df['Z_value'])

pivotted= df.pivot('Y_value','X_value','Z_value')

sns.heatmap(pivotted,cmap='coolwarm')
sns.set(rc={'figure.figsize':(30, 30)})
```





9)Discuss the Solution

As one can see solution is symmetric and value of temperature increases from 0 to 1 as we move from outer boundary to inner boundary. One can also see that as we come near inner boundary jump in values of temperature is high. So if someone wants more detailed variation map of temperature near inner boundary, they need to make more number meshes.

Link to the code can be found here

https://colab.research.google.com/drive/1KMljLQ_AiRI1lgMpW4yojvOLnRA3UzS7?usp=sharing