Assignment - 2

FEM and CFD Theory

ME3180

Name: Harshit Shambharkar

Roll No: ME21BTECH11019

2D unsteady heat conduction equation

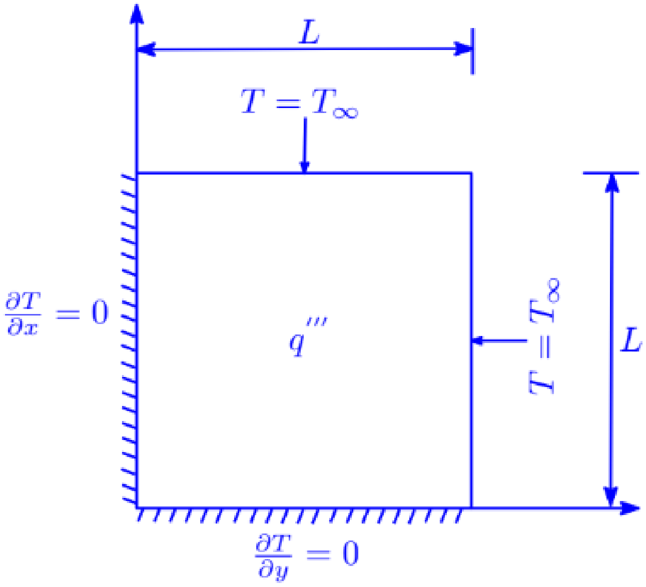
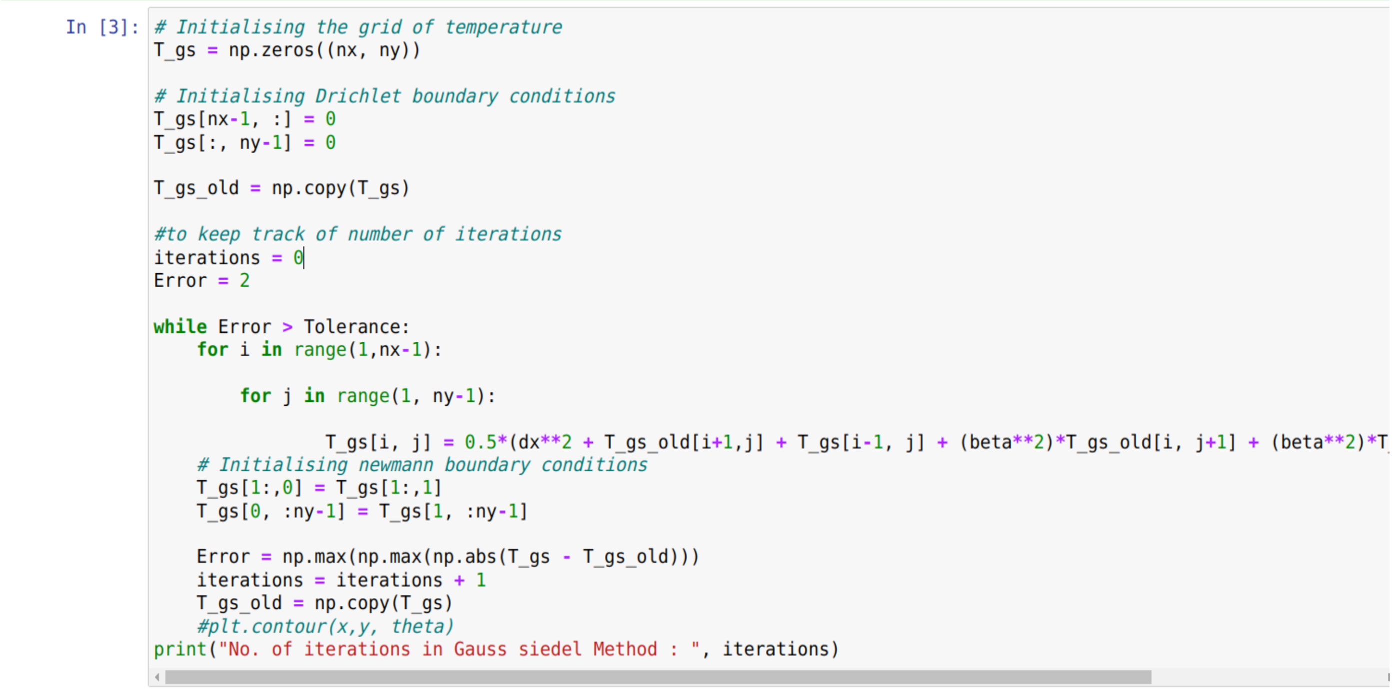

L
$T = T_\infty$
$\frac{\partial T}{\partial x} = 0$
q'''
$T = T_\infty$
L
$\frac{\partial T}{\partial y} = 0$

Figure 1: Computational Domain

Below snapshots of different methods are attached

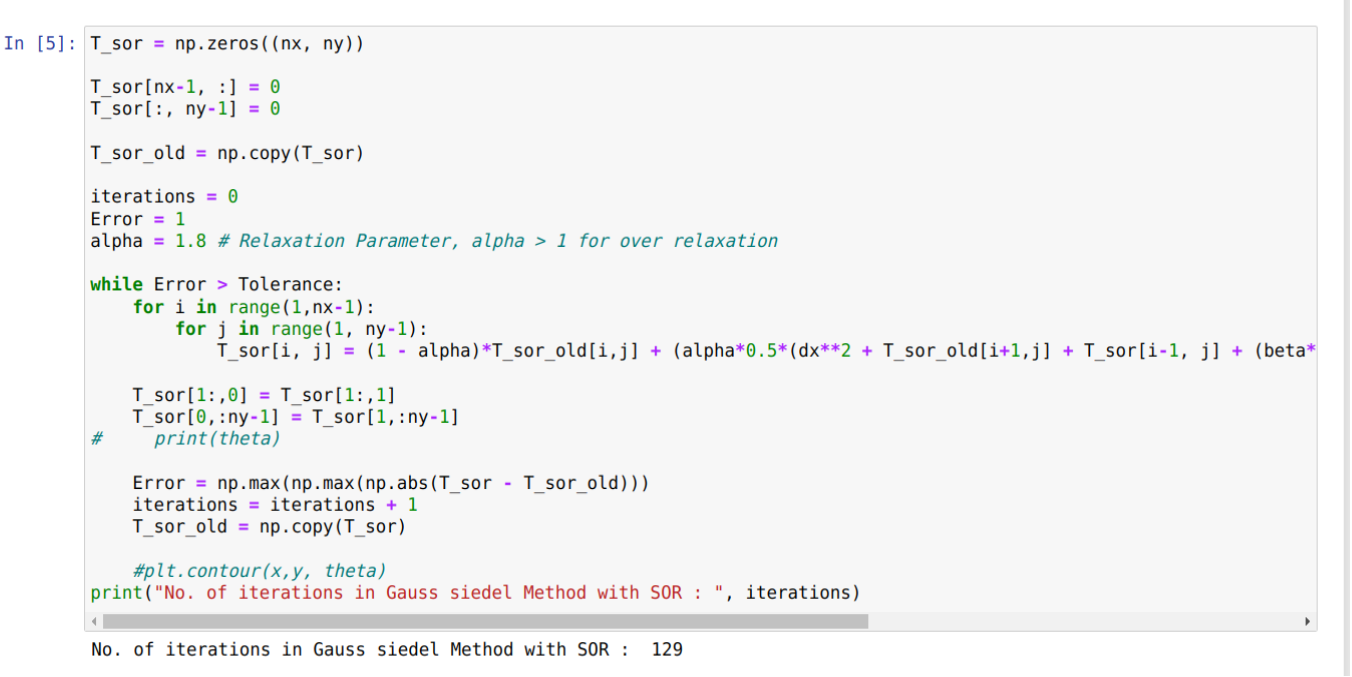
Gauss Siedel



```
In [3]: # Initialising the grid of temperature
        T_gs = np.zeros((nx, ny))

        # Initialising Drichlet boundary conditions
        T_gs[nx-1, :] = 0
        T_gs[:, ny-1] = 0

        T_gs_old = np.copy(T_gs)

        #to keep track of number of iterations
        iterations = 0
        Error = 2

        while Error > Tolerance:
            for i in range(1,nx-1):

                for j in range(1, ny-1):

                        T_gs[i, j] = 0.5*(dx**2 + T_gs_old[i+1,j] + T_gs[i-1, j] + (beta**2)*T_gs_old[i, j+1] + (beta**2)*T_
            # Initialising newmann boundary conditions
            T_gs[1:,0] = T_gs[1:,1]
            T_gs[0, :ny-1] = T_gs[1, :ny-1]

            Error = np.max(np.max(np.abs(T_gs - T_gs_old)))
            iterations = iterations + 1
            T_gs_old = np.copy(T_gs)
            #plt.contour(x,y, theta)
        print("No. of iterations in Gauss siedel Method : ", iterations)
```

No. of iterations in Gauss Siedel are: 490

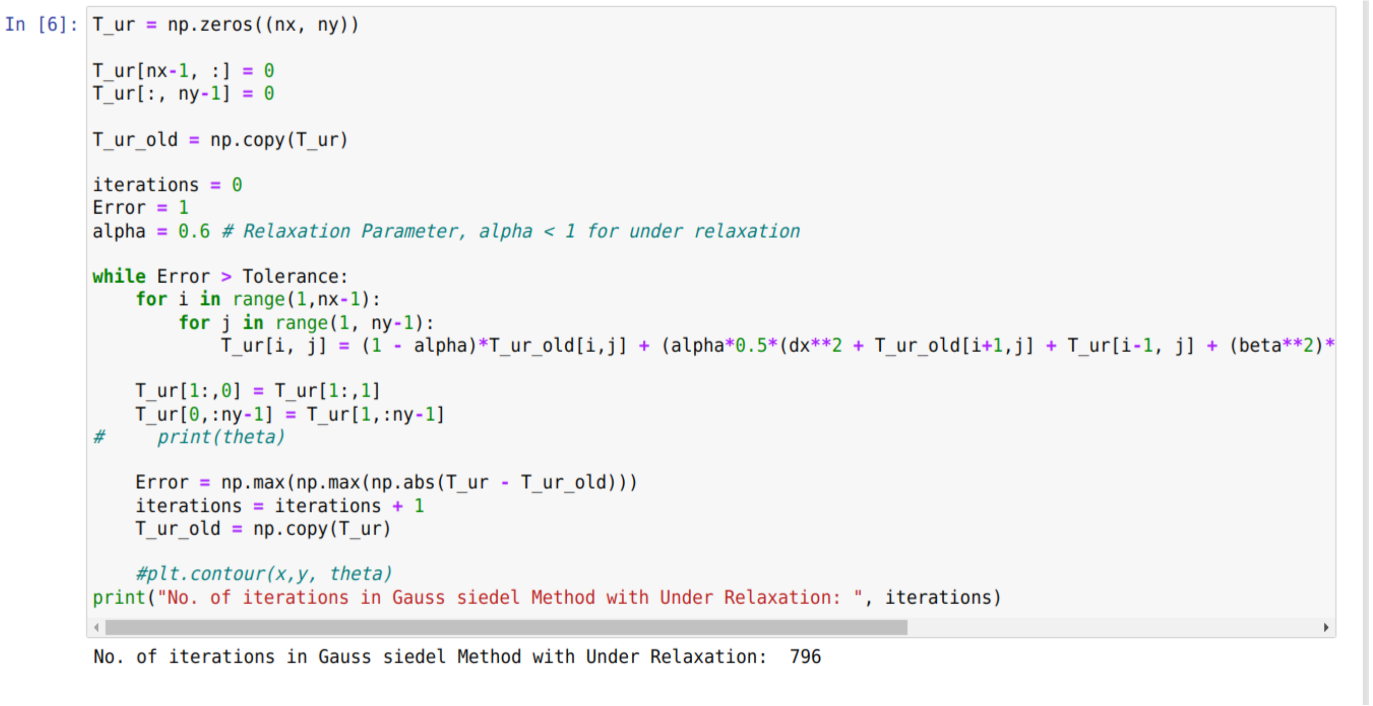
Gauss Siedel with Over Relaxation



```
In [5]: T_sor = np.zeros((nx, ny))

        T_sor[nx-1, :] = 0
        T_sor[:, ny-1] = 0

        T_sor_old = np.copy(T_sor)

        iterations = 0
        Error = 1
        alpha = 1.8 # Relaxation Parameter, alpha > 1 for over relaxation

        while Error > Tolerance:
            for i in range(1,nx-1):
                for j in range(1, ny-1):
                    T_sor[i, j] = (1 - alpha)*T_sor_old[i,j] + (alpha*0.5*(dx**2 + T_sor_old[i+1,j] + T_sor[i-1, j] + (beta*

            T_sor[1:,0] = T_sor[1:,1]
            T_sor[0,:ny-1] = T_sor[1,:ny-1]
        #     print(theta)

            Error = np.max(np.max(np.abs(T_sor - T_sor_old)))
            iterations = iterations + 1
            T_sor_old = np.copy(T_sor)

            #plt.contour(x,y, theta)
        print("No. of iterations in Gauss siedel Method with SOR : ", iterations)

        No. of iterations in Gauss siedel Method with SOR :  129
```

No. of iterations in gauss seidel with over relaxation are: 129

Gauss Siedel with Under Relaxation

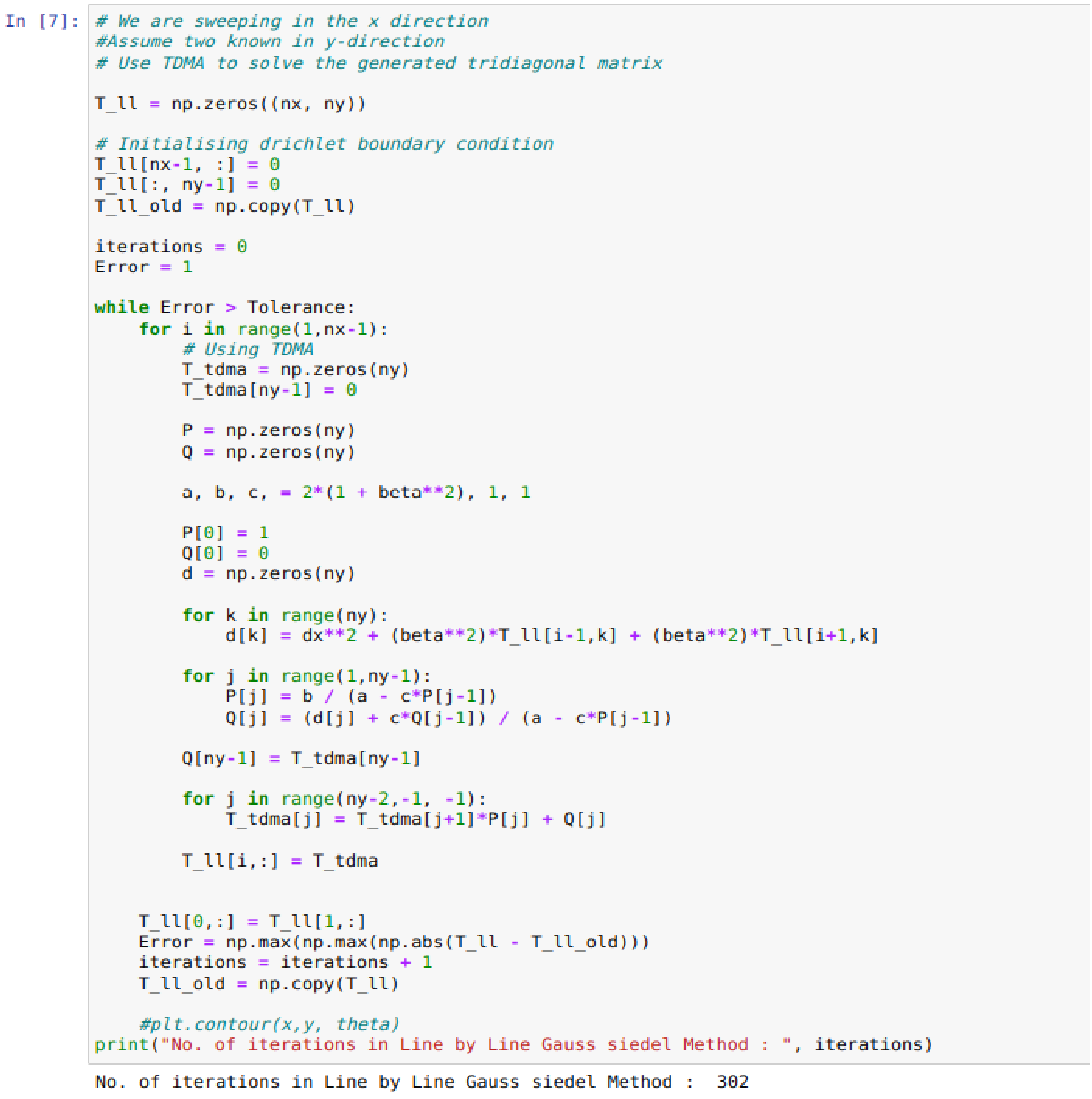


```
In [6]: T_ur = np.zeros((nx, ny))

        T_ur[nx-1, :] = 0
        T_ur[:, ny-1] = 0

        T_ur_old = np.copy(T_ur)

        iterations = 0
        Error = 1
        alpha = 0.6 # Relaxation Parameter, alpha < 1 for under relaxation

        while Error > Tolerance:
            for i in range(1,nx-1):
                for j in range(1, ny-1):
                    T_ur[i, j] = (1 - alpha)*T_ur_old[i,j] + (alpha*0.5*(dx**2 + T_ur_old[i+1,j] + T_ur[i-1, j] + (beta**2)*

            T_ur[1:,0] = T_ur[1:,1]
            T_ur[0,:ny-1] = T_ur[1,:ny-1]
        #     print(theta)

            Error = np.max(np.max(np.abs(T_ur - T_ur_old)))
            iterations = iterations + 1
            T_ur_old = np.copy(T_ur)

            #plt.contour(x,y, theta)
        print("No. of iterations in Gauss siedel Method with Under Relaxation: ", iterations)
```

```
No. of iterations in Gauss siedel Method with Under Relaxation:  796
```

No. of iterations in Gauss Siedel with Under Relaxation are: 796

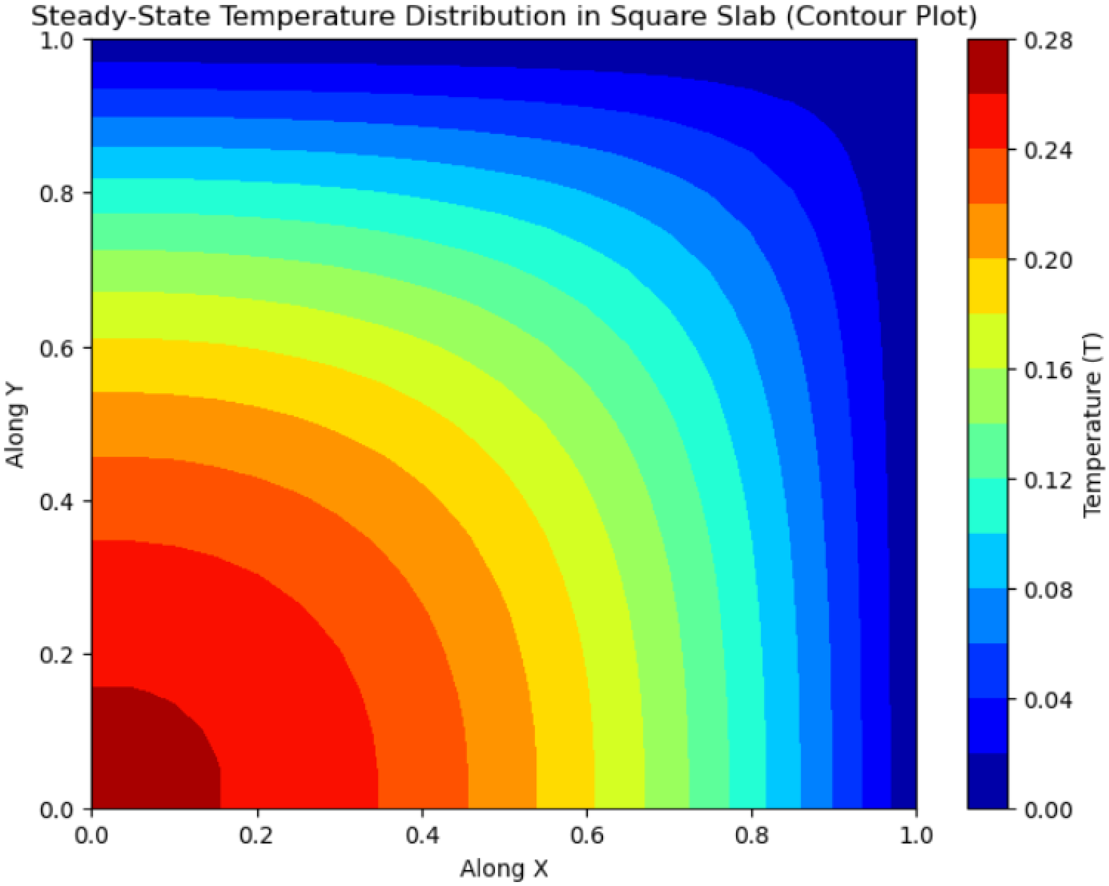
Line by Line Gauss Siedel



```
In [7]: # We are sweeping in the x direction
        #Assume two known in y-direction
        # Use TDMA to solve the generated tridiagonal matrix

        T_ll = np.zeros((nx, ny))

        # Initialising drichlet boundary condition
        T_ll[nx-1, :] = 0
        T_ll[:, ny-1] = 0
        T_ll_old = np.copy(T_ll)

        iterations = 0
        Error = 1

        while Error > Tolerance:
            for i in range(1,nx-1):
                # Using TDMA
                T_tdma = np.zeros(ny)
                T_tdma[ny-1] = 0

                P = np.zeros(ny)
                Q = np.zeros(ny)

                a, b, c, = 2*(1 + beta**2), 1, 1

                P[0] = 1
                Q[0] = 0
                d = np.zeros(ny)

                for k in range(ny):
                    d[k] = dx**2 + (beta**2)*T_ll[i-1,k] + (beta**2)*T_ll[i+1,k]

                for j in range(1,ny-1):
                    P[j] = b / (a - c*P[j-1])
                    Q[j] = (d[j] + c*Q[j-1]) / (a - c*P[j-1])

                Q[ny-1] = T_tdma[ny-1]

                for j in range(ny-2,-1, -1):
                    T_tdma[j] = T_tdma[j+1]*P[j] + Q[j]

                T_ll[i,:] = T_tdma


            T_ll[0,:] = T_ll[1,:]
            Error = np.max(np.max(np.abs(T_ll - T_ll_old)))
            iterations = iterations + 1
            T_ll_old = np.copy(T_ll)

            #plt.contour(x,y, theta)
        print("No. of iterations in Line by Line Gauss siedel Method : ", iterations)

        No. of iterations in Line by Line Gauss siedel Method :  302
```

Number of iterations in Line by Line Gauss Siedel are: 302

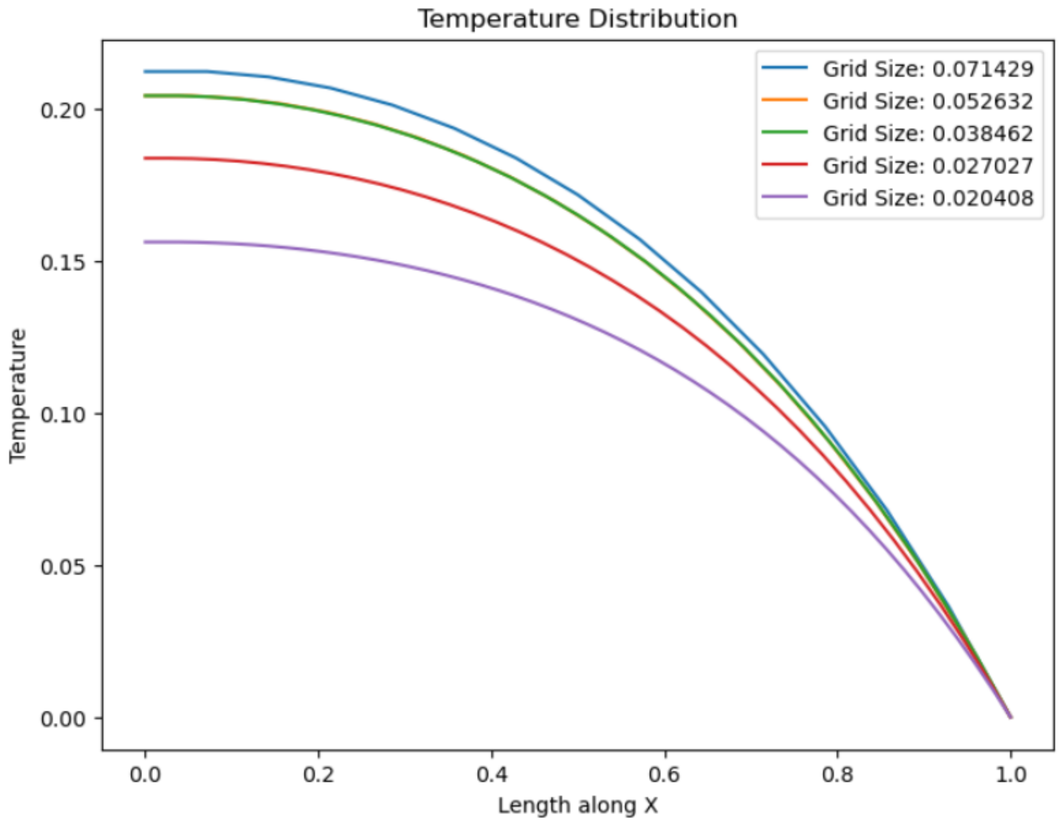
The contour of temperature distribution is plotted below



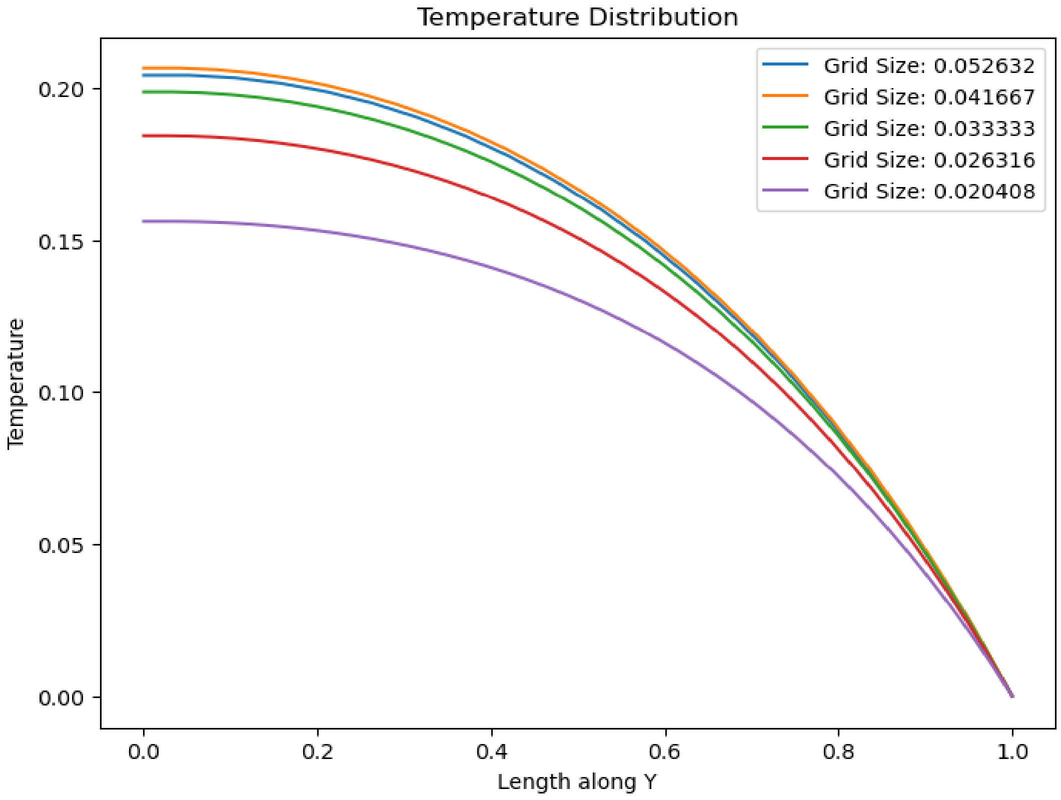
Grid Independence test check

The centre line temperature distribution along x and y is performed for different grid sizes is plotted

Temperature distribution along X:



Temperature distribution along Y:


Temperature Distribution
Grid Size: 0.052632
Grid Size: 0.041667
Grid Size: 0.033333
Grid Size: 0.026316
Grid Size: 0.020408
Temperature
Length along Y