Applied Computational Methods in Mechanical Sciences (ME466)

Assignment 9

Himanshu Kumar

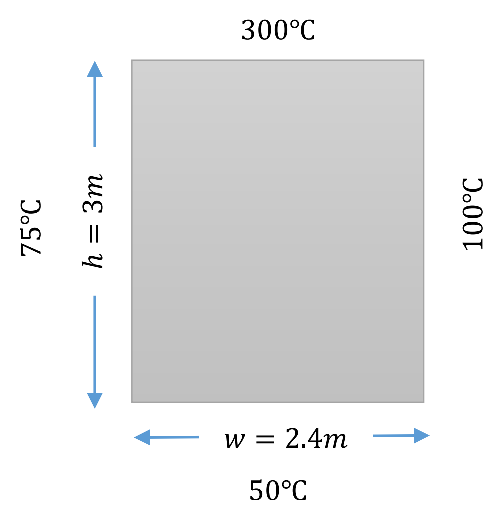
16ME234

October 28, 2019

Problem Statement:

A plate of 2.4m x 3.0m is subjected to temperature as shown in the figure. Using suitable uniform grid, find the temperature distribution throughout the interior of the plate. It is governed by the following differential equation:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$



Python Code:

```
import time

import matplotlib.pyplot as plt

from mpl_toolkits import mplot3d

import numpy as np
```

```
def disp(x):
    for i in range(len(x)):
        print(x[i])


def uniform_grid(lims, n):
    rng = lims[1]-lims[0]
    delx=rng/n

    xm = [0 for i in range(n+1)]
    for i in range(n):
        xm[i+1] = xm[i]+delx

    return(xm,delx)


def matrix_form_solve_plot(lx,nx,ly,ny,err_lim):
    global x,dx,y,dy
    x,dx = uniform_grid(lx,nx)
    y,dy = uniform_grid(ly,ny)

    e= 1
    c= -2*(1+( (dx*dx)/(dy*dy) ))
    w= 1

    s= (dx*dx)/(dy*dy)
    n= (dx*dx)/(dy*dy)

    lx = len(x)-1
    ly = len(y)-1

    global t
    t = [ [60 for i in range(lx+1)] for j in range(ly+1) ]
    err = [ [0 for i in range(lx+1)] for j in range(ly+1) ]
```

```
Qi =0

# boundary conditions
# 1. left
for i in range(ly+1):
    t[i][0] = 75
# 2. Right
for i in range(ly+1):
    t[i][lx] = 100
# 3. Top
for i in range(lx+1):
    t[ly][i] = 300
# 4. Bottom
for i in range(lx+1):
    t[0][i] = 50

pt = [ [60 for i in range(lx+1)] for j in range(ly+1) ]
for i in range(0,ly+1):
    for j in range(0,lx+1):
        pt[i][j] = t[i][j]

#gauss run
run = 0
while(1):

    for i in range(1,ly):
        for j in range(1,lx):
            t[i][j] = (Qi - ( s*t[i-1][j] + n*t[i+1][j] + e*t[i][j+1]+ w*t[i][j-1] ))/(c)
   # omega = 1.6
   # for i in range(1,ly):
    #     for j in range(1,lx):
    #         t[i][j] =(1-omega)*t[i][j] +  omega*(Qi - ( s*t[i-1][j] + n*t[i+1][j] +
e*t[i][j+1]+ w*t[i][j-1] ))/(c)
```

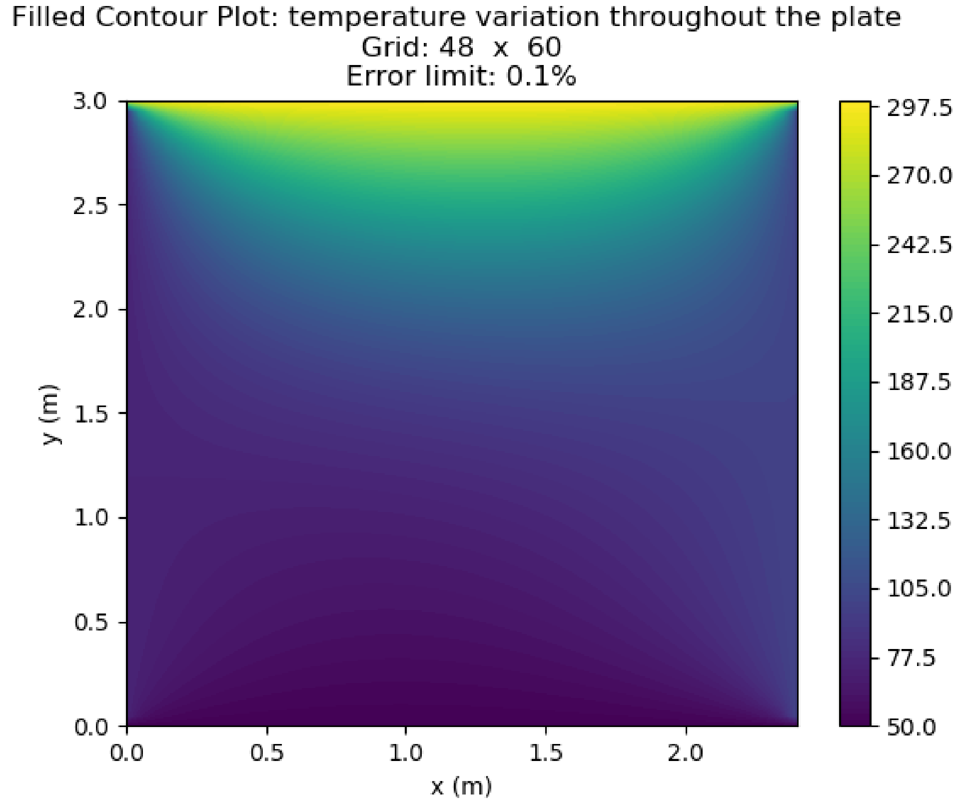
```
    #Error calculation
    for i in range(1,ly):
        for j in range(1,lx):
            try:
                err[i][j] = abs((t[i][j]- pt[i][j])/t[i][j])
            except:
                pass
    #finding maximum error:
    max_err = 0
    for i in range(1,ly):
        for j in range(1,lx):
            if(err[i][j]>max_err):
                max_err = err[i][j]

    if(max_err<err_lim):
        break

    #reiterate
    run = run+1
    for i in range(1,ly):
        for j in range(1,lx):
            pt[i][j]= t[i][j]

print("\n No. of iterations:",run)
print ("\n CPU time: ", time.process_time(),'s')

#plotting filled contour
X, Y = np.meshgrid(x, y)
fig,ax=plt.subplots(1,1)
cp = ax.contourf(X, Y, t,100,cmap = 'viridis')
fig.colorbar(cp) # Add a colorbar to a plot
```

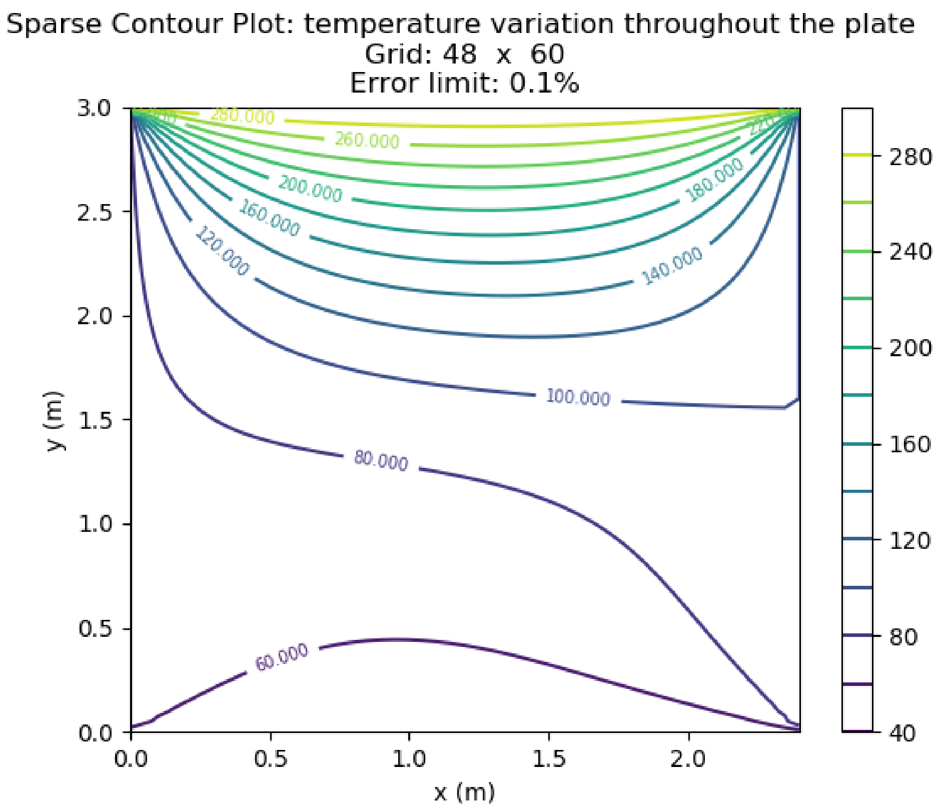
```
title = 'Grid: '+str(nx)+'  x  '+str(ny)+'\nError limit: '+str(err_lim*100)+'%'
ax.set_title('Filled Contour Plot: temperature variation throughout the plate \n'+title)
ax.set_xlabel('x (m)')
ax.set_ylabel('y (m)')
plt.show()



#plotting line contour
X, Y = np.meshgrid(x, y)
fig,ax=plt.subplots(1,1)
cp = ax.contour(X, Y, t,15)
fig.colorbar(cp) # Add a colorbar to a plot
ax.clabel(cp, inline=1, fontsize=7)
ax.set_title('Sparse Contour Plot: temperature variation throughout the plate \n'+title)
ax.set_xlabel('x (m)')
ax.set_ylabel('y (m)')
plt.show()

#plotting detailed line contour
X, Y = np.meshgrid(x, y)
fig,ax=plt.subplots(1,1)
cp = ax.contour(X, Y, t,100)
fig.colorbar(cp) # Add a colorbar to a plot
ax.clabel(cp, inline=1, fontsize=7)
ax.set_title('Dense Contours Plot: temperature variation throughout the plate \n'+title)
ax.set_xlabel('x (m)')
ax.set_ylabel('y (m)')
plt.show()



#plotting 3D
X, Y = np.meshgrid(x, y)
```

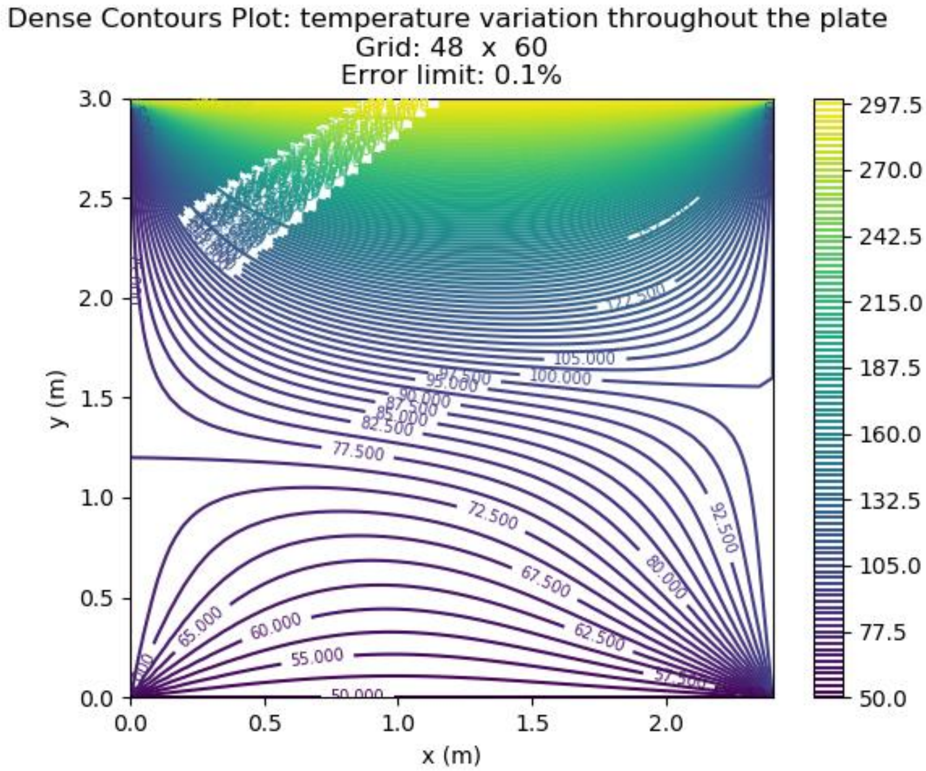
```
    fig = plt.figure()
    ax = plt.axes(projection='3d')
    ax.contour3D(X, Y, t, 50, cmap='viridis')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('t')
    ax.set_title('3D contour')
    plt.show()


mag = 1
nx=48*mag
ny=60*mag
err_lim = 0.001
limx = (0,2.4)
limy = (0,3.0)

matrix_form_solve_plot(limx,nx,limy,ny,err_lim)
```

Results:

1. <u>**Grid 48 x 60 ; Error limit = 0.1% ; Gauss-Siedel**</u>

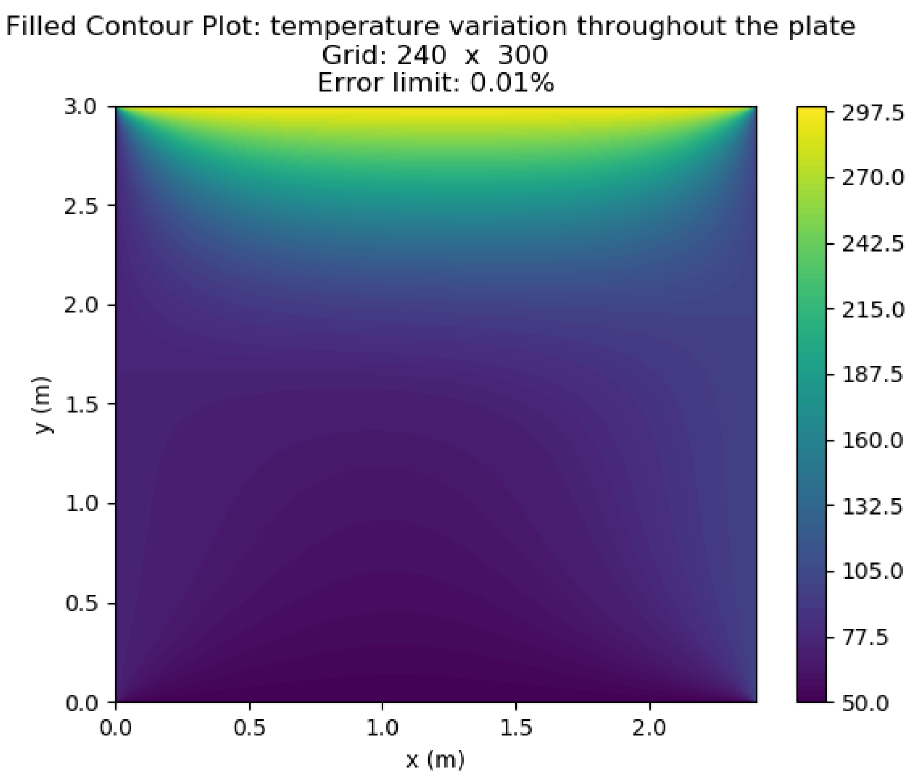


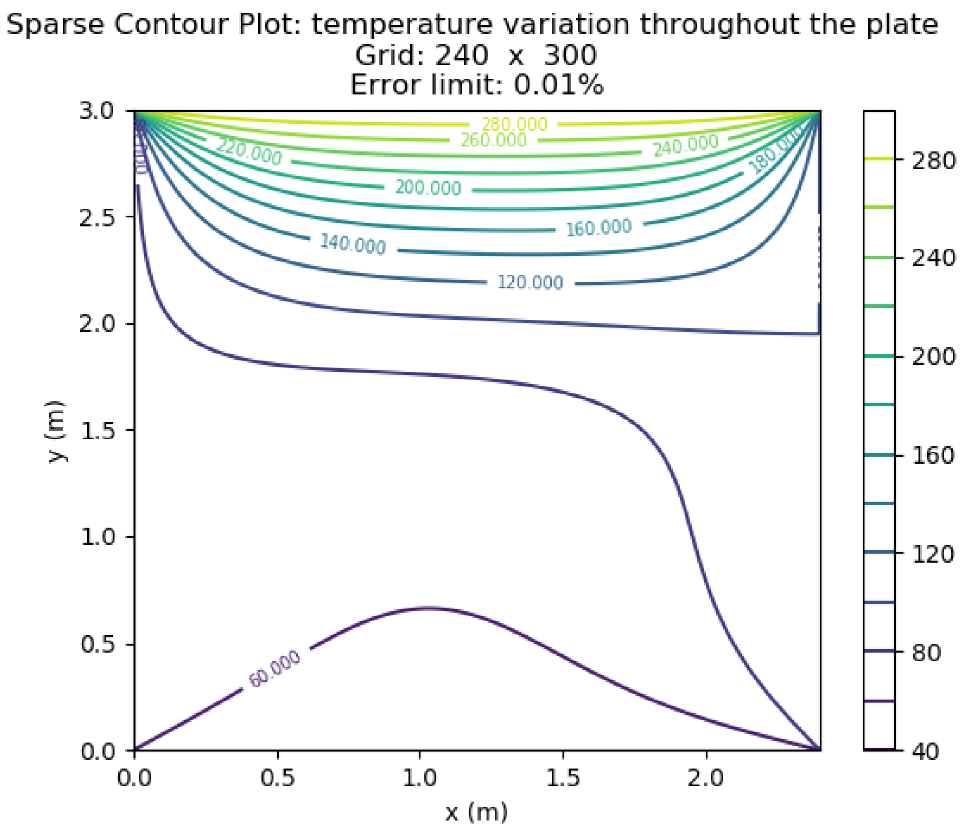


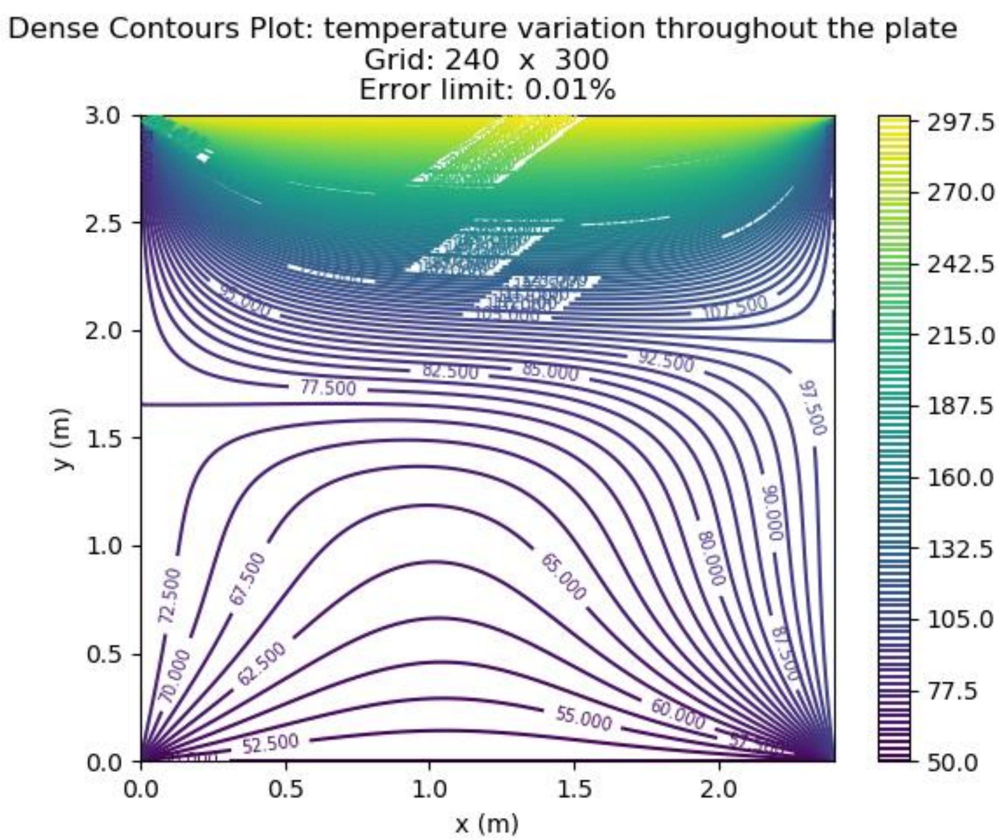



```
No. of iterations: 387

 CPU time:  2.15625 s
```

2. Grid 240 x 300 ; Error limit = 0.01% ; Gauss-Siedel

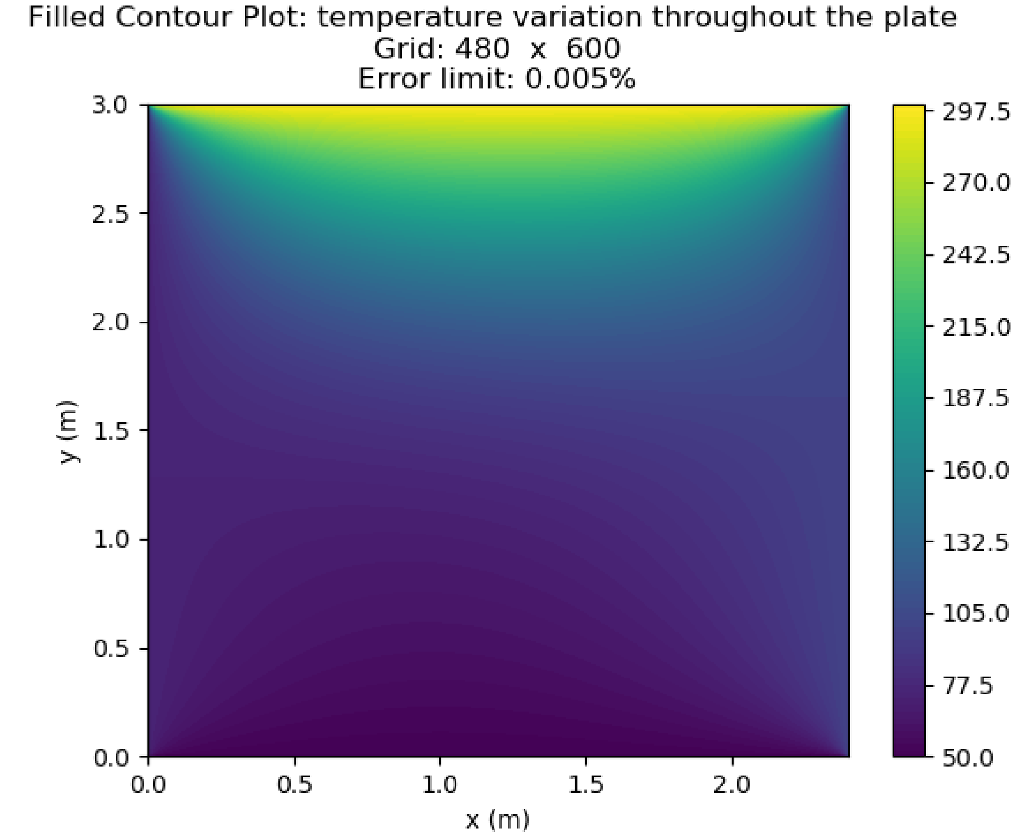


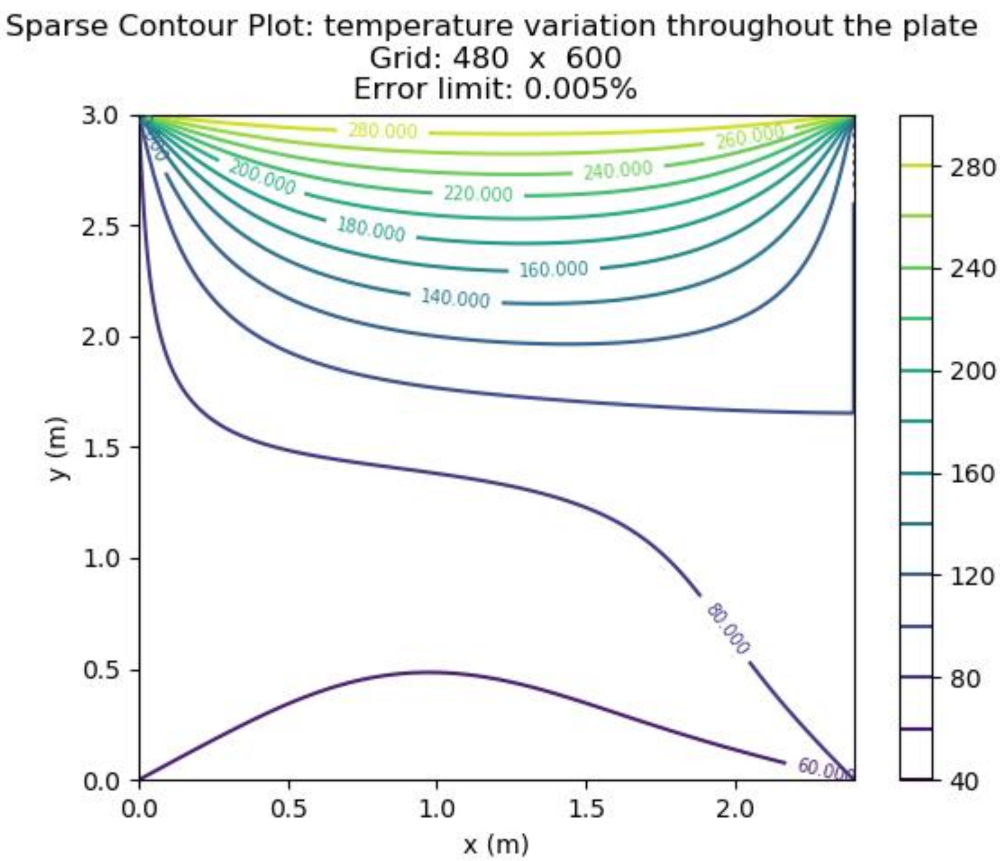


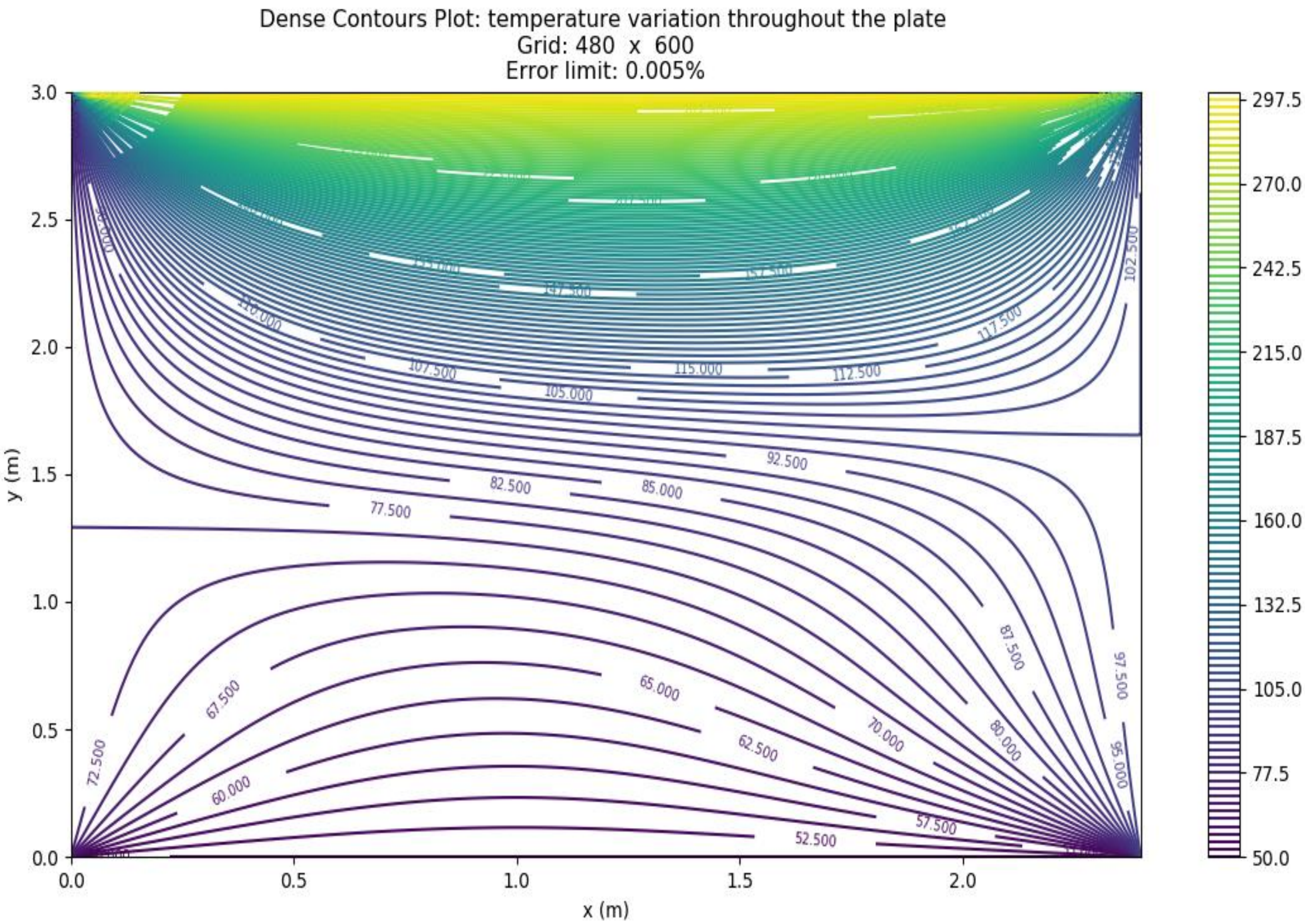



```
No. of iterations: 4936

 CPU time:  500.671875 s
```

3. <u>Grid 480 x 600 ; Error limit = 0.005% ; Successive-over relaxation, $\boldsymbol{\omega = 1.6}$</u>








```
No. of iterations: 8248

 CPU time:  3997.546875 s
```