EE2703: Assignment 5

KORRA SRIKANTH
EE19B033

April 22, 2021

Introduction

We wish to solve for the currents in a resistor. The currents depend on the shape of the resistor. we also want to know which part of the resistor is likely to get hottest.

Aim:

The goal of this assignment is the following:

- To solve for currents in a system.
- To solve 2-D Laplace equations in an iterative manner.
- To understand how to vectorize code in python.
- To plot graphs to understand the 2-D Laplace equation.

Task 1:
Defining parameters and Initializing the potential matrix:

- Asking the user to pass parameter values N_x , N_y and r.
- Define the Parameters, The parameter values taken for my particular code were N_x and N_y and No of iterations : 1500
- These values are taken to discuss about Stopping condition,etc
- To allocate the potential array $\phi = 0$ .Note that the array should have N_y rows and N_x columns.
- To find the indices which lie inside the circle of radius(r) using meshgrid() by equation :

$$X^2 + Y^2 \leq r^2 \tag{1}$$

- Then assign 1v to those indices.
- To plot a contour plot of potential ϕ and to mark V=1 region in red

Code:

```
if (len(argv) ==4):
    Nx = int(argv[1])          # No. of steps along the x direction
    Ny = int(argv[2])          # No. of steps along the y direction
    r = float(argv[3])
    print("Using user provided params")
else:
    Nx = 25
    Ny = 25
    r = 8
    print("Using default parameters")

phi = zeros((Nx,Ny))
```

```
Niter = 1500
x = linspace(-Nx/2,Nx/2,Nx)
y = linspace(Nx/2,-Nx/2,Ny)
Y,X = meshgrid(y,x)

# Finding out the points with 1v potential
Z = where(X*X + Y*Y <= r*r )

# Marking potential = 1V points in the phi matrix
for i in range(Z[0].size):
    phi[Z[0][i],Z[1][i]] = 1

# Plotting the 1V points and the contour plot of phi matrix
figure(1)
title('1V Potential and the contour plot of potential matrix')
xlabel(r'$x$')
ylabel(r'$y$')
plot(x[Z[0]],y[Z[1]],'ro', label = '1V Potential')
contour(X,Y,phi)
legend()
grid()
show()
```

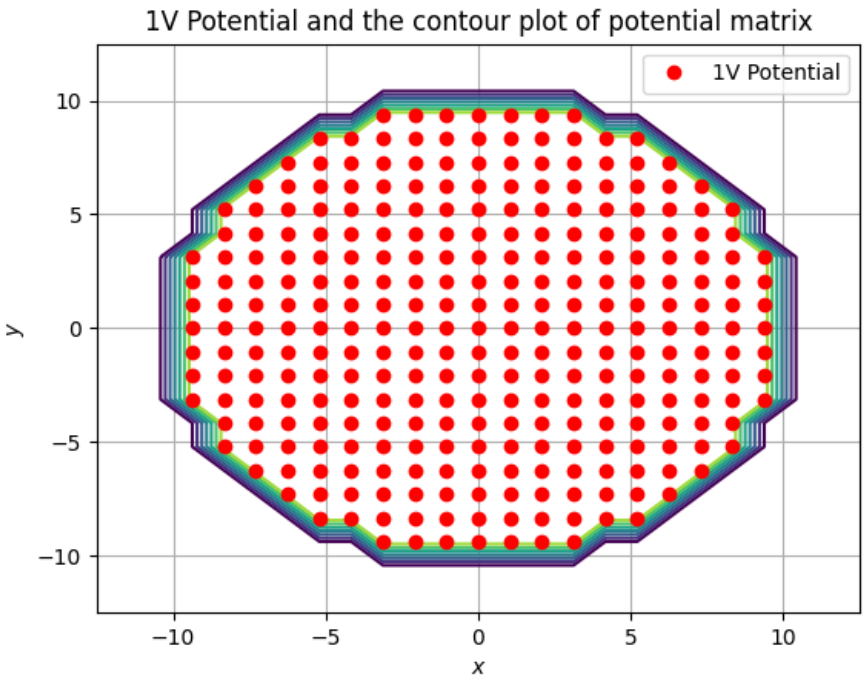

Figure 1: Contour plot of initial potential

Task 2 :
Performing the Iterations:

- To update the potential ϕ according to Equation below using vectorized code

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{4} \tag{2}$$

- To plot the errors in semilog and loglog and observe how the errors are evolving.

Code:

```
# iteration which calculates the steady state potential array
error=[]
e1 = zeros((30))
# iteration which calculates the steady state potential array
for  i in range(Niter) :
    oldphi = phi.copy()
    phi[1:-1,1:-1] = 0.25*(phi[1:-1,0:-2]+phi[1:-1,2:]+phi[0:-2,1:-1]+phi[2:,1:-1])
    phi[1:-1,0]=phi[1:-1,1]
    phi[1:-1,Nx-1]=phi[1:-1,Nx-2]
    phi[0,1:-1]=phi[1,1:-1]
    phi[0,0]=phi[0,1]
    phi[0,Nx-1]=phi[0,Nx-2]
    phi[Ny-1,1:-1]=0.0
    phi[Z]=1.0
    error.append((abs(oldphi-phi)).max())
```

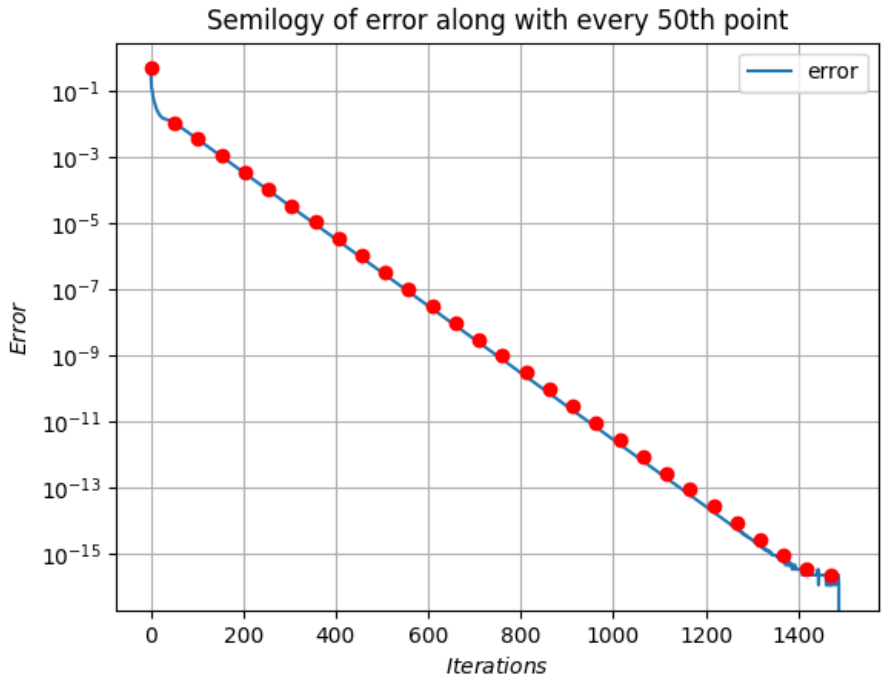

Figure 2: Semilog plots of Error vs No.of Iterations

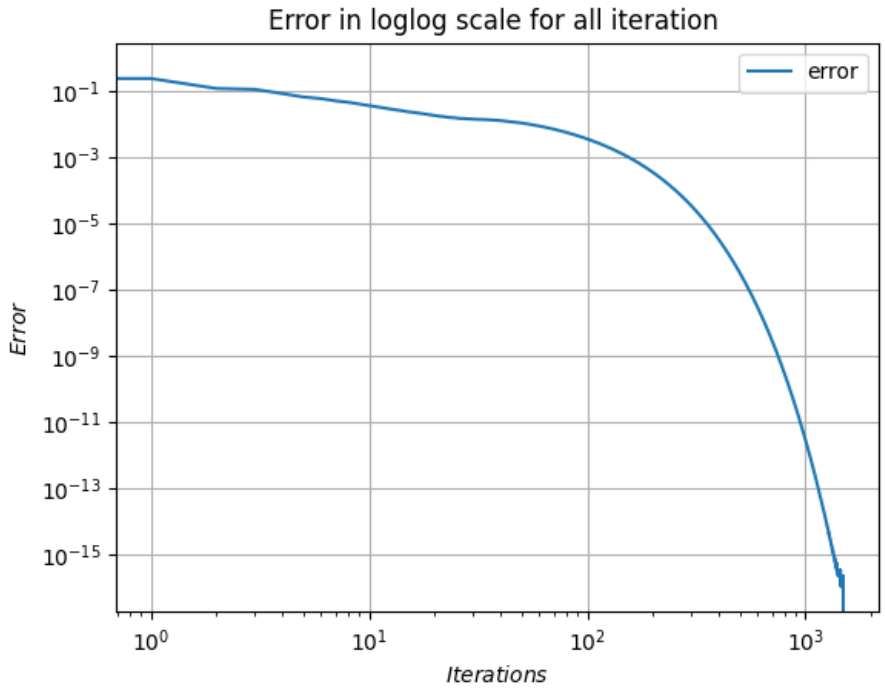

Figure 3: Log-Log plots of Error vs No.of Iterations

Task 3 :
Fitting the exponential function to the error plots:

- To find the fit using Least squares for all iterations and for iterations ≥ 500 separately and compare them.
- As we know that error follows Ae^{Bx} at large iterations, we use equation given below to fit the errors using least squares

$$logy = logA + Bx \tag{3}$$

- To find the time constant of error function obtained for the two cases using lstsq and compare them
- To plot the two fits obtained and observe them

Code:

```
expe = error[500:]
logy = log(expe)
n = linspace(501,Niter,Niter-500)
L1 = ones((Niter-500,2))
L1[:,0]= n

expe1 = error[:]
logy1 = log(expe1)
n1 = linspace(1,Niter,Niter)
R1 = ones((Niter,2))
R1[:,0]= n1

# Fitting the error to an exponential for > 500 iterations
err = linalg.lstsq(L1,logy,rcond=-1)[0]
error_fit =  err[1] + err[0]*n
```

```
# Fitting the error to an exponential
err1= linalg.lstsq(R1,logy1,rcond=-1)[0]
error_fit1 =  err1[1] + err1[0]*n1


# Plotting the error fits
figure(4)
title('Error in semilog scalefor all iterations')
xlabel(r'$Iterations$')
ylabel(r'$Error$')
plot(linspace(1,Niter,Niter),log(error),"g",label = 'errors')
plot(n,error_fit,"ro",label = 'fit error > 500 iter')
plot(n1,error_fit1,"b--",label = 'fit error')
grid()
legend()
show()

# Plotting the 3D surface plot of potential
fig4 = figure(5)
ax = p3.Axes3D(fig4)
title('The 3-D Surface Plot of the potential')
ylabel(r'$GROUND$')
surf = ax.plot_surface(X,Y,phi,rstride=1,cstride=1,cmap = cm.jet,label = 'Potential')
show()
```

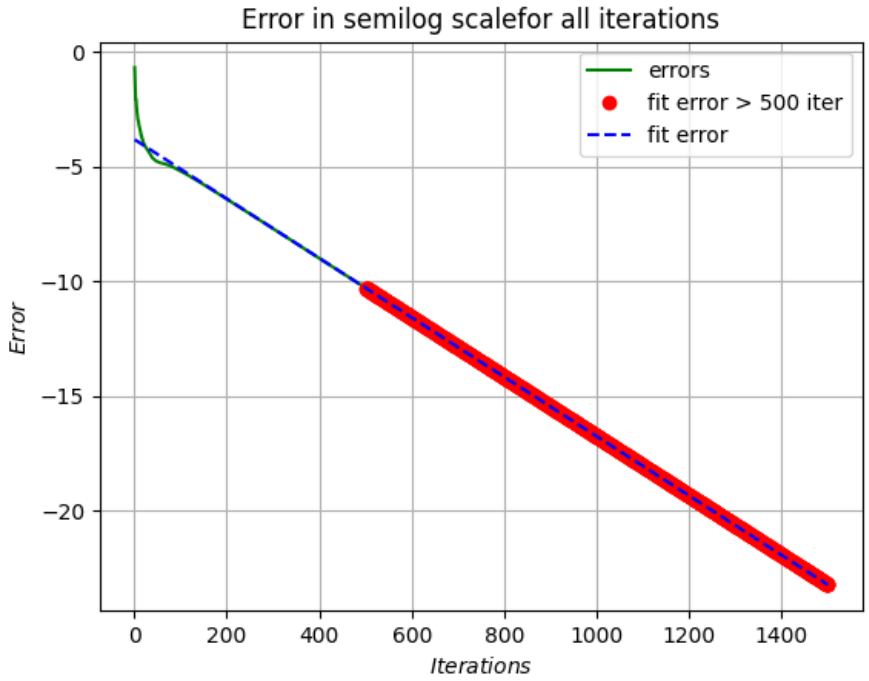

Figure 4: Semilog plot of Error vs No.of Iterations

Task 4 :
Plotting 3D surface plot and Contour plot of potential

- To do a 3-D surface plot of the potential.
- To plot contour plot of potential
- And analyse them and to comment about flow of currents

Code:

```
 # Plotting the 3D surface plot of potential
fig4 = figure(5)
ax = p3.Axes3D(fig4)
title('The 3-D Surface Plot of the potential')
ylabel(r'$GROUND$')
surf = ax.plot_surface(X,Y,phi,rstride=1,cstride=1,cmap = cm.jet,label = 'Potential')
show()

# Contour plot of potential
figure(6)
title('Contour plot of the potential')
xlabel(r'$x$')
ylabel(r'$y$')
plot(x[Z[0]],y[Z[1]],'ro', label = '1V Potential')
clabel(contour(x,y,phi) , inline = True , fontsize = 8)
grid()
legend()
show()
```

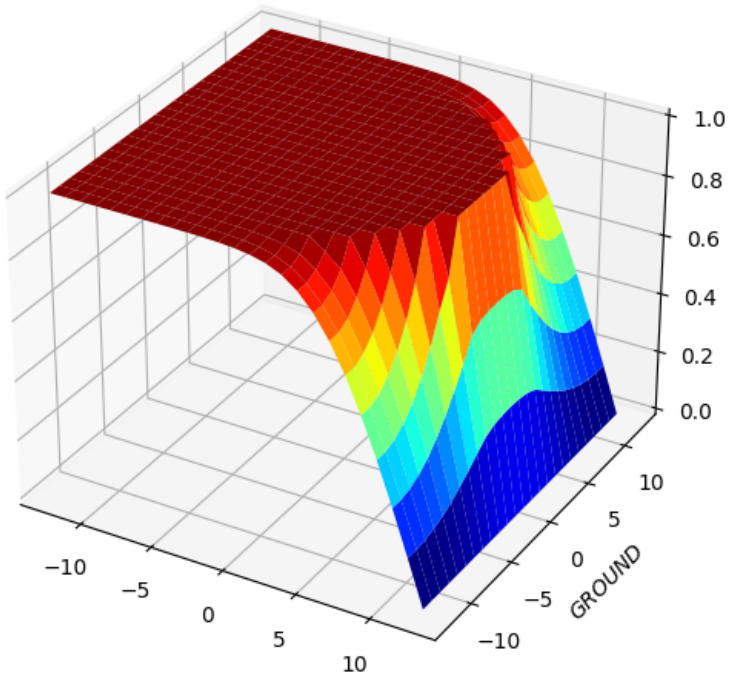

Figure 5: 3-D Surface potential plot

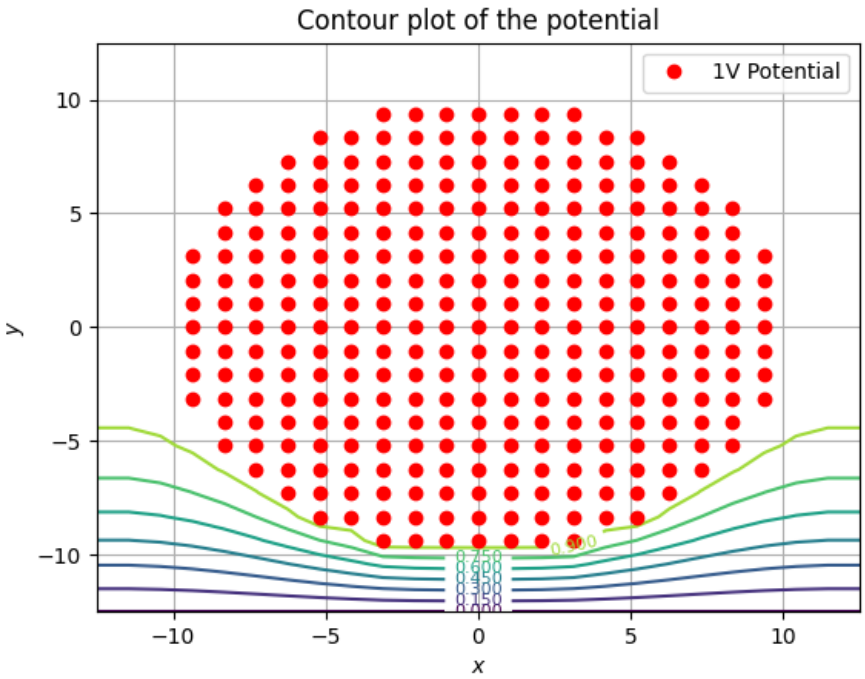

Figure 6: Contour plot of potential

Task 5 :
Vector Plot of Currents:

- To obtain the currents by computing the gradient.
- The actual value of σ does not matter to the shape of the current profile, so we set it to unity. Our equations are

$$J_x = -\frac{\partial \phi}{\partial x} \tag{4}$$

$$J_y = -\frac{\partial \phi}{\partial y} \tag{5}$$

- To program this we use these equations as follows:

$$J_{x,ij} = \frac{1}{2}(\phi_{i,j-1} - \phi_{i,j+1}) \tag{6}$$

$$J_{y,ij} = \frac{1}{2}(\phi_{i-1,j} - \phi_{i+1,j}) \tag{7}$$

Code:

```
# Plotting the current vectors using quiver
figure(7)
title('Quiver plot of the current densities')
xlabel(r'$x$')
ylabel(r'$y$')
plot(x[Z[0]],y[Z[1]],'ro', label = '1V potential')
quiver(y,x,-Jy[::-1,:],-Jx[::-1,:],scale=1.8,scale_units='inches',label="Current density")
grid()
legend()
show()
```

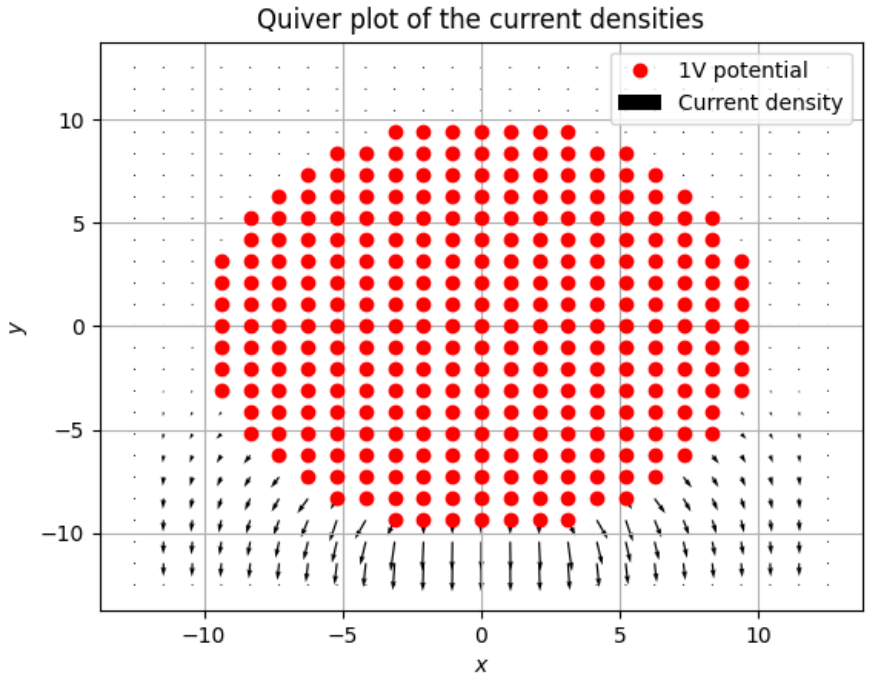

Figure 7: Vector plot of current flow

Additional :

Heat Map of the conductor:

As the current flows in the conductor, it heats up. Thus increasing it's temperature.This phenomenon is called Joule Heating.
The heat equation is given by :

$$\kappa \nabla^2 T = -\frac{1}{\sigma}|j|^2 \tag{8}$$

We take,

$$\kappa = 1, \sigma = 1 \; and \; \Delta x = 1 \; for \; simplicity. \tag{9}$$

Thus expanding this equation gives us:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} + |J|^2}{4(\Delta x)^2} \tag{10}$$

Thus by updating the temperature `Niter` times we get a temperature which converges.
The boundary condition is that at the boundary $\frac{\partial T}{\partial n} = 0$

Code:

```
T = zeros((Nx,Ny))
T[:,:] = 300
sigma = 6*(10**7)
kappa = 385
for i in range(Niter):
    T[1:-1,1:-1] = 0.25*(T[1:-1,0:-2] + T[1:-1,2:] + T[0:-2,1:-1] + T[2:,1:-1] + (((Jx**2)[1:-
    T[1:-1,0]=T[1:-1,1]
    T[1:-1,Nx-1]=T[1:-1,Nx-2]
    T[0,1:-1]=T[1,1:-1]
    T[Z] = 300.0


fig1=figure(4)
ax=p3.Axes3D(fig1)
title('The 3-D surface plot of the temperature')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('Temperature')
ax.plot_surface(X, Y, T, rstride=1, cstride=1, cmap=cm.jet,linewidth=0, antialiased=False)
show()
```

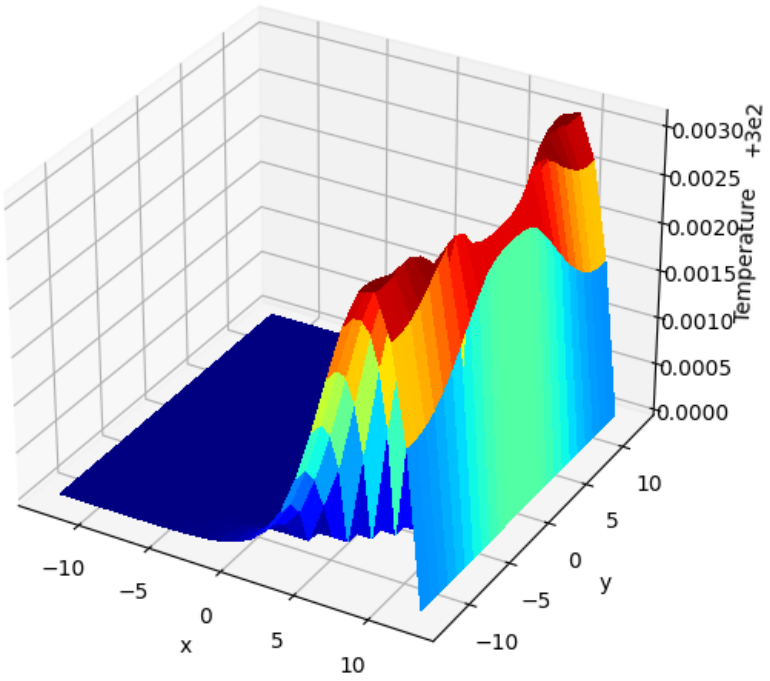

Figure 8: **3D surface plot of the Temperature**

Code:

```
J_sq = Jx**2 + Jy**2

figure(9)
title('Contour plot of the heat generated')
cp = contour(-Y,-X,J_sq)
clabel(cp,inline=True,colors='r')
xlabel('x')
ylabel('y')
grid()
show()
```

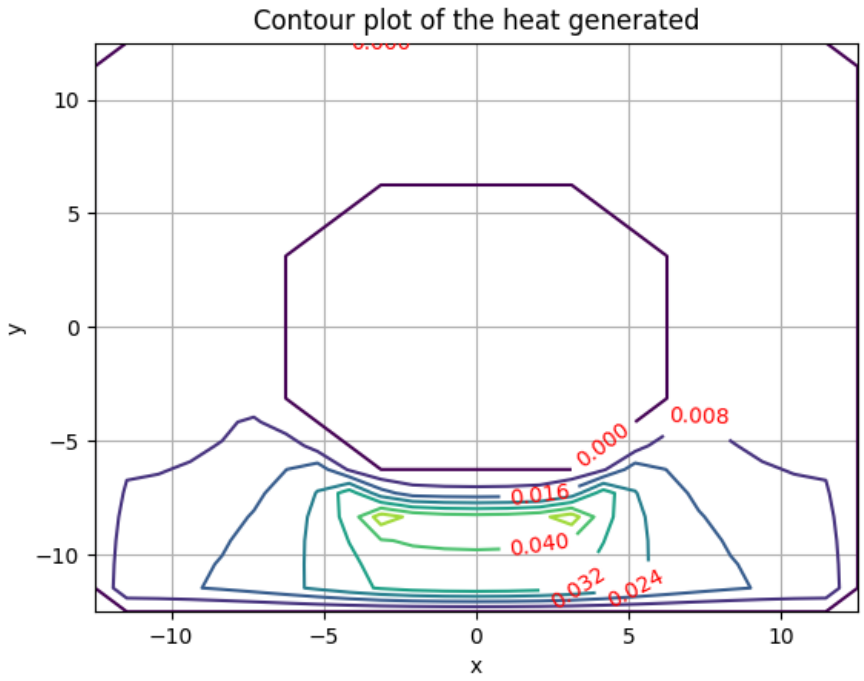

Figure 9: Counter plot of the Heat Generated

Conclusion :

- To conclude , Most of the current is in the narrow region at the bottom.So that is what will get strongly heated.
- Since there is almost no current in the upper region of plate,the bottom part of the plate gets hotter and temperature increases in down region of the plate.
- And we know that heat generated is from $\vec{J}.\vec{E}$ (ohmic loss) so since $\vec{J}$ and $\vec{E}$ are higher in the bottom region of the plate, there will more heat generation and temperature rise will be present.
- So overall we looked the modelling of the currents in resistor in this report ,and we observe that the best method to solve this is to increase N_x and N_y to very high values(100 or $\geq$ 100)and increase the no of iterations too, so that we get accurate answers i.e currents in the resistor.
- But the tradeoff is this method of solving is very slow even though we use vectorized code because the decrease in errors is very slow w.r.t no of iterations.