EE - 2703 Applied Programming Lab

Assignment -5

Laplace Equation

Mohammed Khandwawala EE16B117

Contents

1 Introduction **1**

2 Defining the parameters **1**

3 Allocating potential array and initialization **2**
3.1 Plotting Contour Plot 2

4 Perform the iterations **2**
4.1 Surface Plot of Potential 4
4.2 Contour plot of potential 4
4.3 Vector Plot of Current 4
4.4 Explanation of the graph 8

5 Error Analysis **8**

6 Current Heating **11**
6.1 Explanation for Temperature plot of surface 11

1 Introduction

We wish to solve for the currents in a resistor. A wire is soldered to the middle of a copper plate and its voltage is held at 1 Volt. One side of the plate is grounded, while the remaining are floating. The plate is 1 cm by 1 cm in size. As a result, current flows. The current at each point can be described by a "current density". This current density is related to the local Electric Field by the conductivity:

$$\vec{j} = \sigma \vec{E}$$

Now the Electric field is the gradient of the potential,

$$\vec{E} = -\nabla\phi$$

and continuity of charge yields Combining these equations we obtain

$$\nabla\vec{j} = -\frac{d\rho}{dt}$$

Assuming that our resistor contains a material of constant conductivity, the equation becomes

$$\nabla.(-\sigma\nabla\phi) = -\frac{d\rho}{dt}$$

For DC currents, the right side is zero, and we obtain

$$\nabla^2\phi = 0$$

2 Defining the parameters

Defining grid size (the size of the resistor) , the radius of the conductor and the number of iterations to run for converging the result.

```
# declaring the constants
Nx = 25 # size of the grid
Ny = 25
radius = 8; # radius of the wire
Niter = 1500;
```

3 Allocating potential array and initialization

An array phi is crated with initial value 0 of dimensions Nx by Ny. It will look like

$$\begin{bmatrix} 0 & 0 & 0 & ... & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & \\ 0 & 0 & 0 & ... & 0 & 0 \end{bmatrix} \tag{1}$$

Since we want the origin at the center of the grid. Create two axis from $\frac{-Nx}{2}$ to $\frac{Nx}{2}$ and $\frac{-Ny}{2}$ to $\frac{Ny}{2}$ Using python function meshgrid we can get coordinates for the above to axis array. using the condition that potential inside the conductor is uniform and constant so phi inside the circle should be 1 at all point. $X^2 + Y^2 < 8*8$. Storing these points and using np.where command to make potential 1 in phi grid.

```
# creating vectors x,y for 2-D coordinates
print -Nx/(2*25.0),Ny/(2*25.0)
x = np.linspace(-Nx/(2*25.0),Ny/(2*25.0),25)
y = np.linspace(-Nx/(2*25.0),Ny/(2*25.0),25)

#Y,X coordinates of the grid which we will create
Y,X = np.meshgrid(x,y)

#selecting coordinates inside the circular conductor
ii = np.where(X*X+Y*Y<=(0.35)**2)

#creating grid and marking potential inside the circle 1
phi = np.zeros((Nx,Ny))
phi[ii] = 1
```

3.1 Plotting Contour Plot

```
plt.contour(Y,X,phi)
plt.plot(x[ii[0]],y[ii[1]],'ro')
plt.show()
```

4 Perform the iterations

Now we need to iteratively update potentials for that we need to store old phi matrix , update value , assert boundary conditions and compute error (change). For storing we need to use python copy as normals equals operator does not generate a new phi it will just give it one more reference. Laplace's equation is easily transformed into a difference equation. The equation can be written out in Cartesian coordinates

Assuming ϕ is available at points (x_i,y_i) we can write

$$\frac{d\phi}{dx}_{(x_i,y_i)} = \frac{\phi(x_{i+1/2},y_j)\phi(x_{i-1/2},y_j)}{\Delta x}$$

and

$$\frac{d^2\phi}{dx^2}_{(x_i,y_i)} = \frac{\phi(x_{i+1},y_j) - 2\phi(x_i,y_j) + \phi(x_{i-1},y_j)}{\Delta x}$$

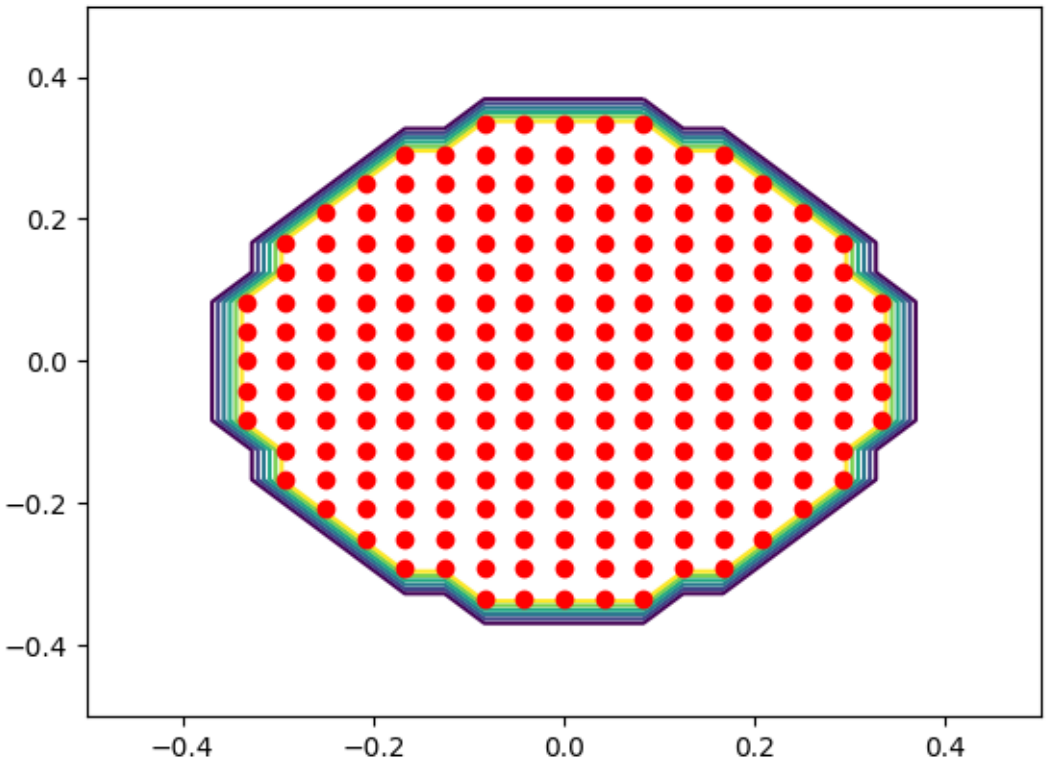
Combining this with the corresponding equation for the y derivatives, we obtain

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

Thus, if the solution holds, the potential at any point should be the average of its neighbors.

Using the result we will update phi matrix by taking the average of the neighbors.

Figure 1: Contour plot for potential of conductor when no current is flowing in the resistor



For Boundary conditions we know that the lower edge is grounded so it is at constant 0 potential. the other three sides we know the current can ot flow outwards so we will make the gradient of potential along each side 0. That is by equating it to value of potential one column before.

For measuring error we will take the maximum of absolute difference between old and new phi matrix.

```
errors = np.zeros(Niter)
for k 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] # Boundary Conditions
        phi[1:-1,-1] = phi[1:-1,-2]
        phi[0,:] = phi[1,:]
        phi[-1,:] = 0 # lower plate potential is 0
        phi[ii] = 1.0
        errors[k] = (abs(phi-oldphi)).max(); #error measurment
```

4.1 Surface Plot of Potential

Plotting the 3-D plot representing potential on each point of the surface.

```
#3-D surface plot of potential
fig1 = plt.figure(1)
ax = p3.Axes3D(fig1)
plt.title('The 3-D surface plot of potential')
surf = ax.plot_surface(-Y,-X,phi.T,rstride = 1,cstride = 1, cmap = plt.cm.jet)
plt.show()
```

4.2 Contour plot of potential

```
#contour plot of potentials
plt.contour(-X,-Y,phi)
plt.plot(y[ii[0]],x[ii[1]],'ro')
plt.show()
```

4.3 Vector Plot of Current

Now to obtaint the current we need to calculate the gradient of the potential.

$$j_x = \frac{-d\phi}{dx}$$

$$j_y = \frac{-d\phi}{dy}$$

This numerically translates to

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

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

vector plots can plotted using quiver function.

```
#vector plot of the electric field
Jx = np.zeros((Nx,Ny))
Jy = np.zeros((Nx,Ny))
Jx[1:-1,1:-1] = 0.5*(phi[1:-1,0:-2]-phi[1:-1,2:])
Jy[1:-1,1:-1] = 0.5*(phi[0:-2,1:-1]-phi[2:,1:-1])
plt.quiver(y,x,-Jx[::-1,:],-Jy[::-1,:])
plt.plot(y[ii[0]],x[ii[1]],'ro')
plt.show()
```

Figure 2: 3-D plot representing voltage at different points

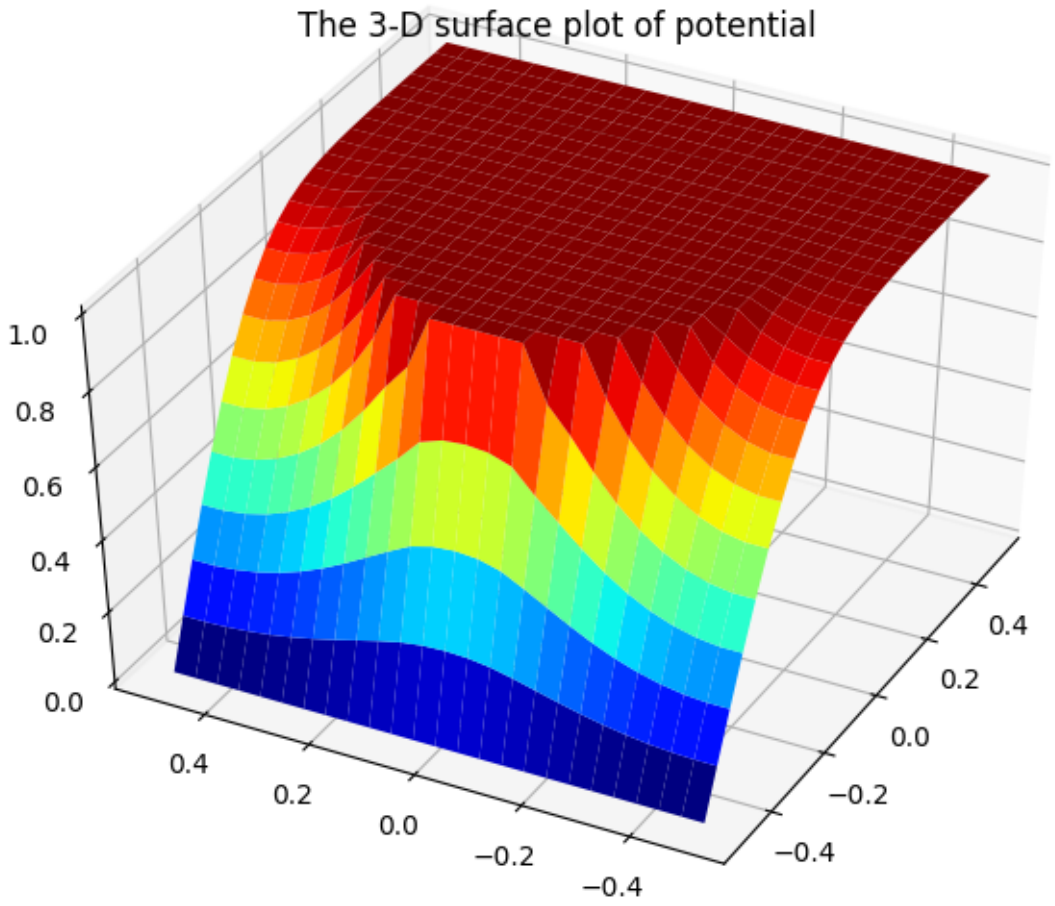


Figure 3:

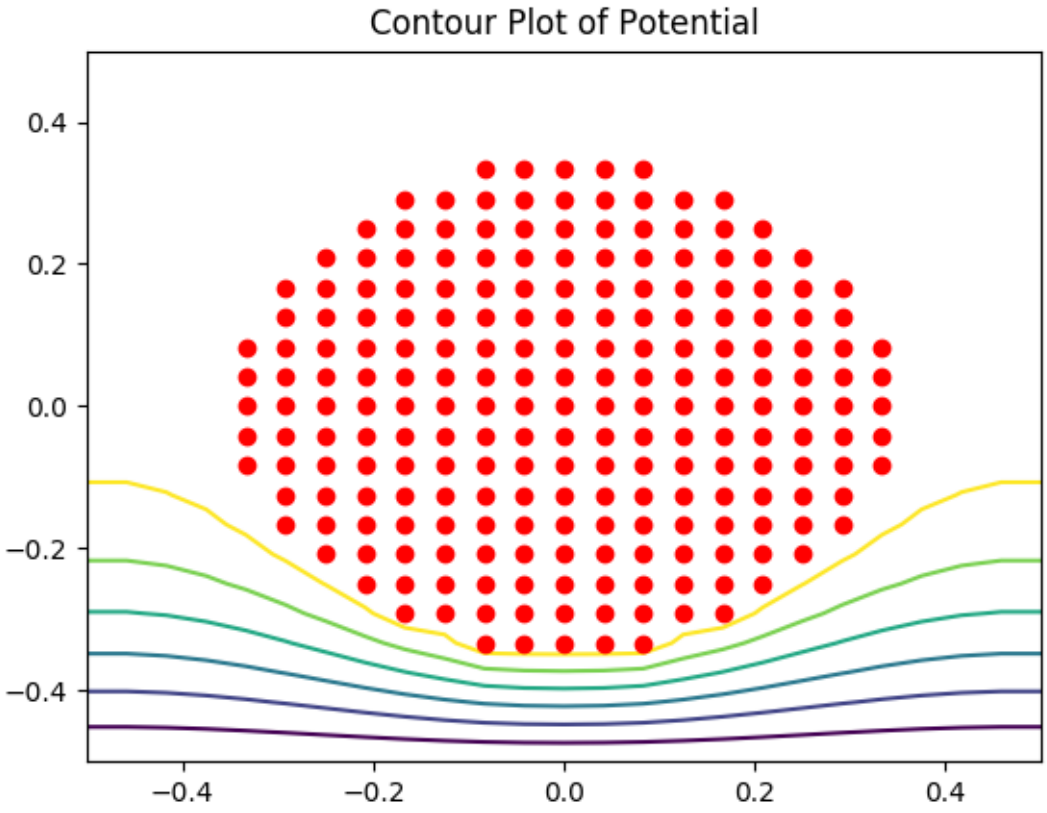


Figure 4:

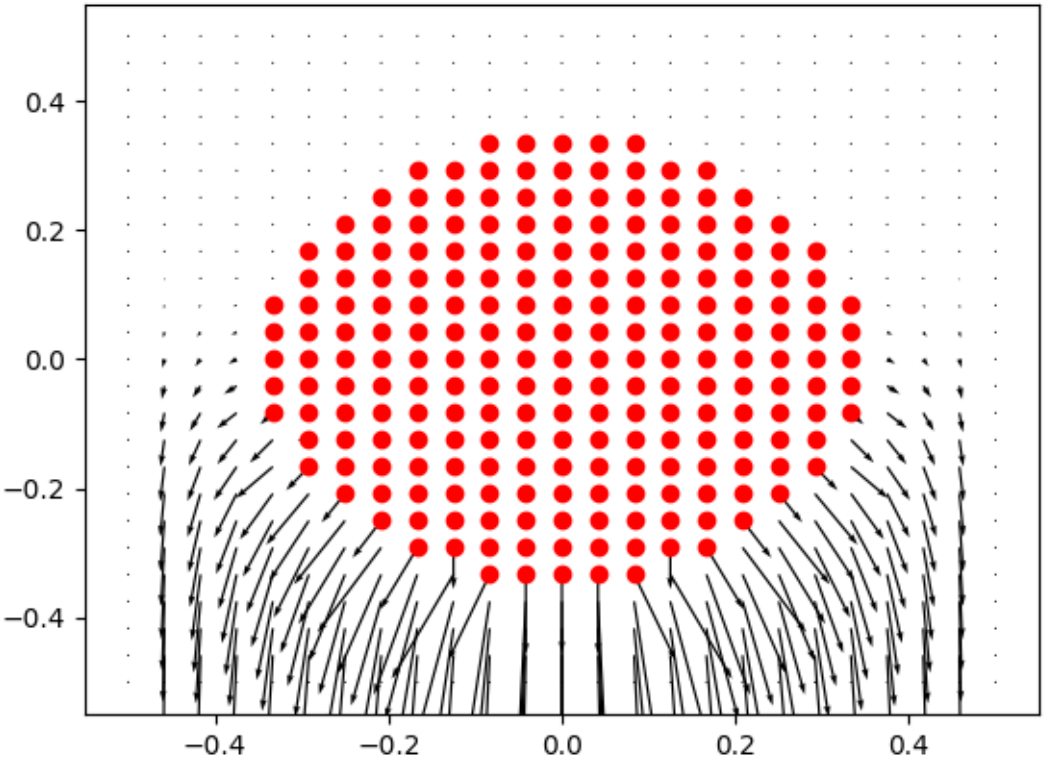


Figure 5:

4.4 Explanation of the graph

As seen from contour plot of potential ,current obtained here is perpendicular to equipotential. And the current negligible in the other half because the current flow through the path of least resistance . The path between the 1V conductor and ground in through lower half of the plate and thus no current flows through the upper half.

5 Error Analysis

```
plt.semilogy(errors)
plt.label("Semi-log plot of Error vs iterations")
plt.show()
```

This semilog plot of error shows straight line with negative slope. Error in this case falls as Ae^{B*k} whrere k is number of iterations and A,B are constants. So error can can be written as

$$log(error) = log(A) + B * k$$

With the error obtained we can fit it to the straight line to obtain constants A and B. We sample error at 50 points and then obtain a fit . fitting complete data. In second cane we will take error only after 500 iterations.

```
#fitting error model
#creating matrix of coeff
A = np.zeros((Niter/50,2))
A[:,0] = 1
A[:,1] = np.log(np.arange(1,Niter+1,50))
```

```
#finding unknown variable to fit error
fit_1 = linalg.lstsq(A,np.log(errors50))[0]

#fitting error model excluding first 500 points
A = np.zeros(((Niter-500)/50,2))
A[:,0] = 1
A[:,1] = np.log(np.arange(501,Niter+1,50))
fit_2 = linalg.lstsq(A,np.log(errors50[10:]))[0]

#printing coeffs obtained by both the model
print fit_1
print fit_2
```

Now we will plot the above two cases along with actual error plot and see which one data gave a better fit. Clearly from the plots (See Figure 8 and 9) it is clear that removing first 500 iteration gives a better fit. The difference is ore clear with less iterations at higher valve all seem to converge.

```
c = np.ones(Niter)
xiter = np.arange(1,Niter+1)
fit_1_v = np.exp(c*fit_1[0] + (xiter)*fit_1[1])
fit_2_v = np.exp(c*fit_2[0] + (xiter)*fit_2[1])

plt.semilogy(xiter[::50],errors[::50],"ro")
plt.semilogy(xiter[::50],fit_1_v[::50],"g^")
plt.semilogy(xiter[::50],fit_2_v[::50],"b+")
plt.xlabel("Interations")
plt.ylabel("Log(Error)")
plt.title("Plot of log of error and model fitting of error for Niter="+str(Niter))
plt.legend(["Orignal error","Model fit (all values)","Model fit (excluding first 500)"
plt.show()
```

$$Error = \sum_{k=Niter+1}^{\infty} error_k$$

$$Error < \sum_{k=Niter+1}^{\infty} Ae^{Bk}$$

$$Error \approx \int_{Niter+0.5}^{\infty} Ae^{Bk} dk$$

$$Error \approx \frac{-A}{B} exp(B(Niter + 0.5))$$

From the above result if we calculate Error by substituting values we obtained.Maximum error from the result and and the last per iteration change in phi (last value of error) do not match. Since error falls as Ae^{Bk} , $\frac{1}{B}$ is the half life. Error obtained times halflife will give us correct value which seems to match in the output.

```
#for 700 iterations
A    0.0241262089052
B   -0.014197697826
Max error = -A/B exp(B(Niter + 1)) 8.08942957309e-05
Max Error times half life -1.14851276663e-06
Error in last step 1.16492469504e-06

#for 1500 iterrations
A    0.0241282489
B   -0.0141978380383
Max error = -A/B exp(B(Niter + 1)) 9.44220600578e-10
Max Error times half life -1.34058911594e-11
Error in last step 1.35974564941e-11
```

Figure 6:

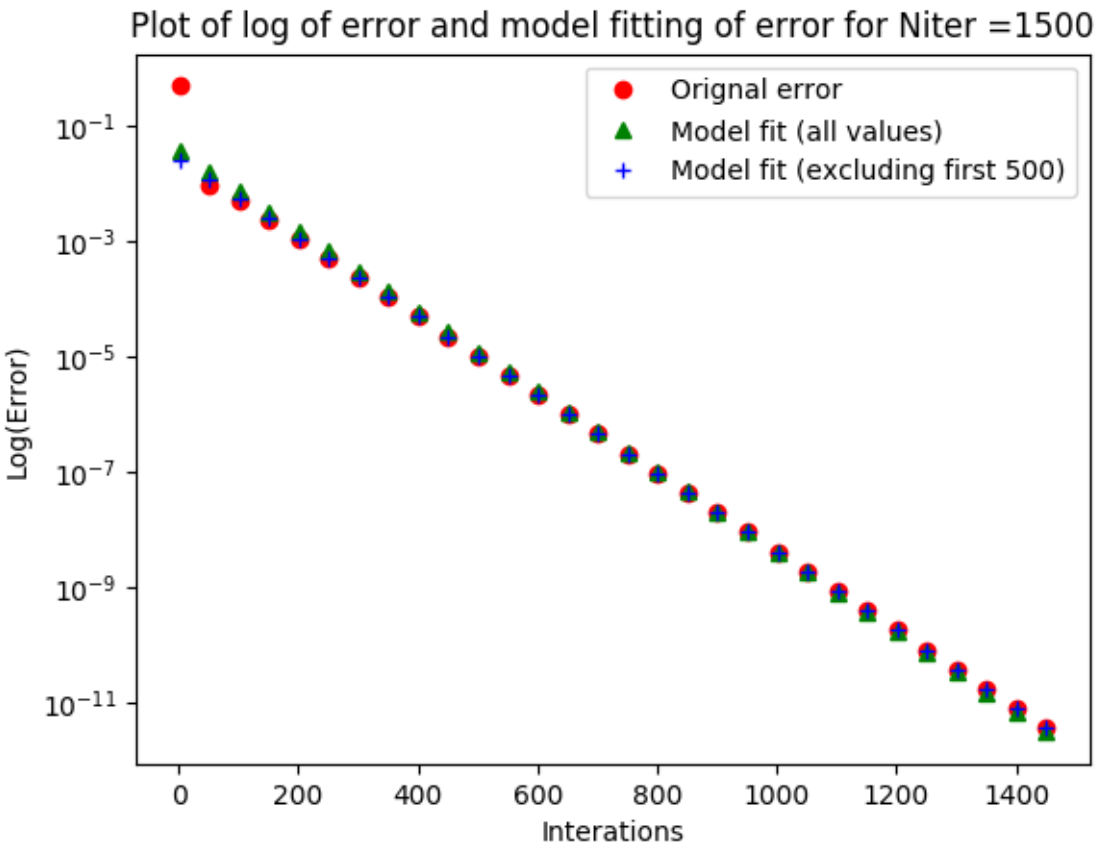
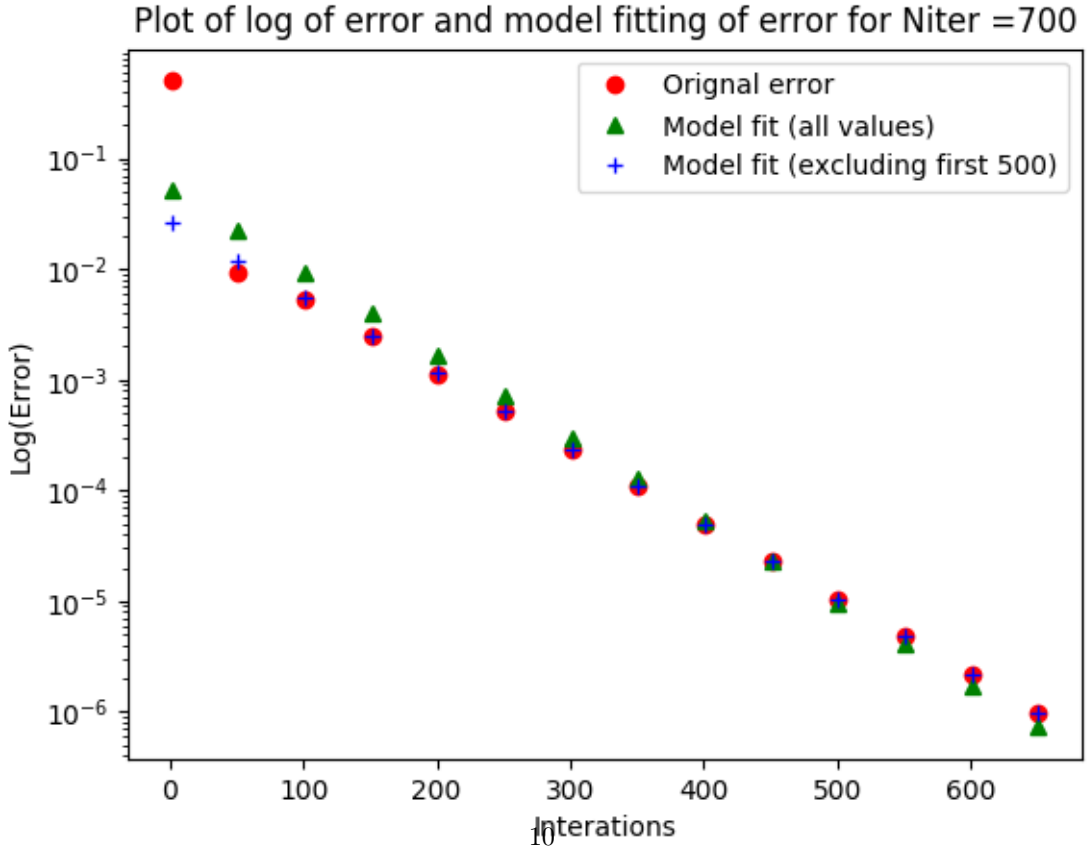


Figure 7:



6 Current Heating

From Current value from the previous part we can calculate temperature at all points to know the heating effect. It can be done from the equation below.

$$\nabla.(k\nabla T) = -\frac{1}{\sigma} \mid J \mid^2$$

Solving this similarly as we did for potential

$$\frac{dT}{dx}_{(x_i,y_i)} = \frac{T(x_{i+1/2}, y_j) - T(x_{i-1/2}, y_j)}{\Delta x}$$

and

$$\frac{d^2T}{dx^2}_{(x_i,y_i)} = \frac{T(x_{i+1}, y_j) - 2T(x_i, y_j) + T(x_{i-1}, y_j)}{\Delta x}$$

From the above equation we can obtain the steady state temperature at any point as

$$T_{i,j} = \frac{T_{i-1,j} + T_{i,j-1} + T_{i,j+1} + T_{i+1,j} + \frac{\Delta x^2}{k\sigma} \mid J^2_{i,j} \mid}{4}$$

With the given initial conditions given , wire and the grounded side at 300. And $\frac{dT}{dn} = 0$ (change in T along normal component at all other sides is 0). See Figure 8.

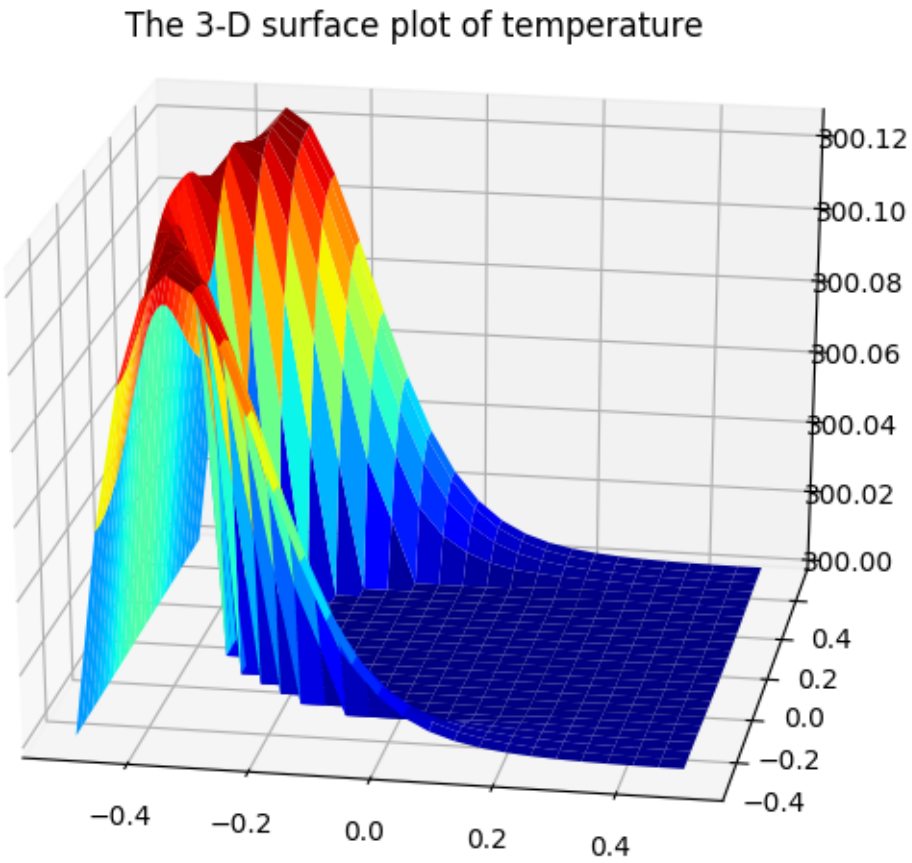
```
dx = Nx/24
temp = np.zeros((Nx,Ny))
temp[ii] = 300
temp[-1,:] = 300
for k in range(Niter):
        #specifying initial conditions
        temp[1:-1,1:-1] = 0.25*(temp[1:-1,0:-2]+temp[1:-1,2:]+temp[0:-2,1:-1]+temp[2:,
        temp[1:-1,0] = temp[1:-1,1] # Boundary Conditions
        temp[0,:] = temp[1,:]
        temp[1:-1,-1] = temp[1:-1,-2]
        temp[-1,:] = 300
        temp[ii] = 300

fig1 = plt.figure(1)
ax = p3.Axes3D(fig1)
plt.title('The 3-D surface plot of temperature')
surf = ax.plot_surface(-Y,-X,temp.T,rstride = 1,cstride = 1, cmap = plt.cm.jet)
plt.show()
```

6.1 Explanation for Temperature plot of surface

As seen from Figure 8 . The resistor heats up more at the point where current flowing is more. The upper half of the plate remain at 300 as no current flows there.

Figure 8:


The 3-D surface plot of temperature
300.12
300.10
300.08
300.06
300.04
300.02
300.00
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4