Assignment 5: The Resistor Problem

ROHIT KUMAR [EE20B111]

March 11, 2021

Abstract

The main goals of this assignment are :-

- To show how currents depend on shape of resistor and the current's dependency on the shape of the resistor and also which part of the resistor is likely to get hottest.
- To solve 2-D laplace equations using numerical approximation over a grid. how to vectorize in python code. To obtain the potential in the region by solving laplace's equation in 2-D and to plot graphs.

1 Introduction

We find the currents in the resistor after applying boundary conditions and analyse the vector plot of current flow and conclude which part of resistor will become hot. Resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.1In this assignment we intend to find the current density and potential gradient in a resistor.

A cylindrical 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.
We shall use these equations:
The Continuity Equation:

$$\nabla \cdot \vec{j} = -\frac{\partial \rho}{\partial t} \tag{1}$$

Ohms Law:

$$\vec{j} = \sigma \vec{E} \tag{2}$$

The above equations along with the definition of potential as the negative gradient of Field give:

$$\nabla^2 \phi = \frac{1}{\rho}\frac{\partial \rho}{\partial t} \tag{3}$$

For DC Currents, RHS of equation (3) is 0. Hence:

$$\nabla^2 \phi = 0 \tag{4}$$

- Here we use a 2-D plate so the Numerical solutions in 2D can be easily transformed into a difference equation. The equation can be written out in

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{5}$$

$$\frac{\partial \phi}{\partial x}_{(x_i,y_j)} = \frac{\phi(x_{i+1/2}, y_j) - \phi(x_{i-1/2}, y_j)}{\Delta x} \tag{6}$$

$$\frac{\partial^2 \phi}{\partial x^2}_{(x_i,y_j)} = \frac{\phi(x_{i+1}, y_j) - 2\phi(x_i, y_j) + \phi(x_{i-1}, y_j)}{(\Delta x)^2} \tag{7}$$

- Using above equations we get

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{4} \tag{8}$$

- the solution at any point is a sum of values at neighbouring points. The algorithm implemented makes use of the above equation to update ϕ over many iterations till it converges within an acceptable error.

2 Defining Parameters and polential Array

- Define the Parameters, The parameter values taken for my particular code were $N_x = 25$ and $N_y = 25$ and number of iterations : 1500
- Allocate the potential array as $\phi = 0$ .Note that the array should have N_y rows and N_x columns.
- To find the indices which lie inside the circle of radius 0.35 using meshgrid() by equation :

$$X^2 + Y^2 \leq 0.35^2 \tag{9}$$

- Then assign 1 V to those indices.

The python code snippet is as shown:

```
Nx = 25;
Ny = 25;
radius = 8;
Niter = 1500;

# Parameters given through command line
if(len(sys.argv) > 1 & len(sys.argv) < 4):
Nx = int(sys.argv[1])
Ny = int(sys.argv[2])
Niter = int(sys.argv[3])

phi = zeros((Ny, Nx))
x = linspace(-0.5, 0.5, Nx)
y = linspace(-0.5, 0.5, Ny)
n = array(range(Niter))
niter = arange(500, 1500, 0.1)
X, Y = meshgrid(x, -y)

# The coordinates can be found for points inside the given radius.
A = (X*X) + (Y*Y)
ii = where(A <= (0.35*0.35))

# Alloting the value of Phi as 1.0 at those coordinates.
phi[ii] = 1.0
figure(1)
semilogy(n,errors)
semilogy(n[::50], errors[::50], 'ro')
title("Error versus iteration")
xlabel(r'$Iteration\rightarrow$', size = 15)
ylabel(r'$Error\rightarrow$', size = 15)
grid(True)
```

The plot for the initial potential contour is as shown:

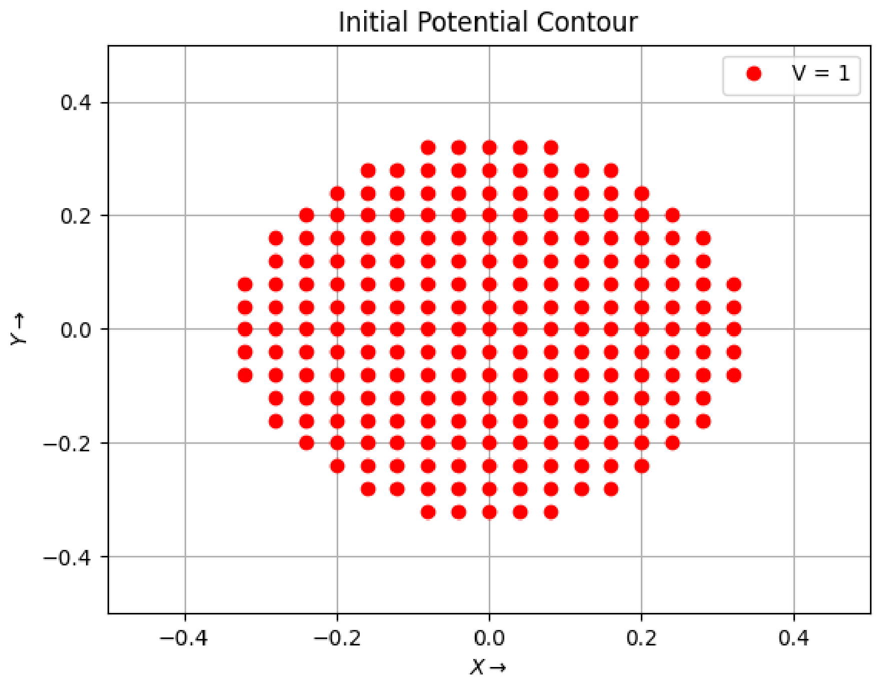


Figure 1: Contour plot of initial potential

3 Performing the Iterations

3.1 Updating Potential Array

- Update the potential ϕ according to Equation below using vectorized code

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{4} \tag{10}$$

- To apply Boundary Conditions where there is no electrode, the gradient of ϕ should be tangential. This is implemented by Equation given below , basically potential should not vary in the normal direction so we equate the last but row or column to outermost row or column correspondingly when applying boundary conditions for a side of plate,implemented using Vectorized code

$$\frac{\partial \phi}{\partial n} = 0 \tag{11}$$

3.2 Applying boundary conditions and Calculating error

The python code is as shown:-

```
errors = empty((Niter, 1))
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:,

# Applying the boundary conditions.
phi[1:-1, 0] = phi[1:-1, 1]
phi[1:-1, -1] = phi[1:-1, -2]
phi[0, 1:-1] = phi[1, 1:-1]
phi[ii] = 1.0
errors[k] = (abs(phi - oldphi)).max()
```

4 The Error Estimations

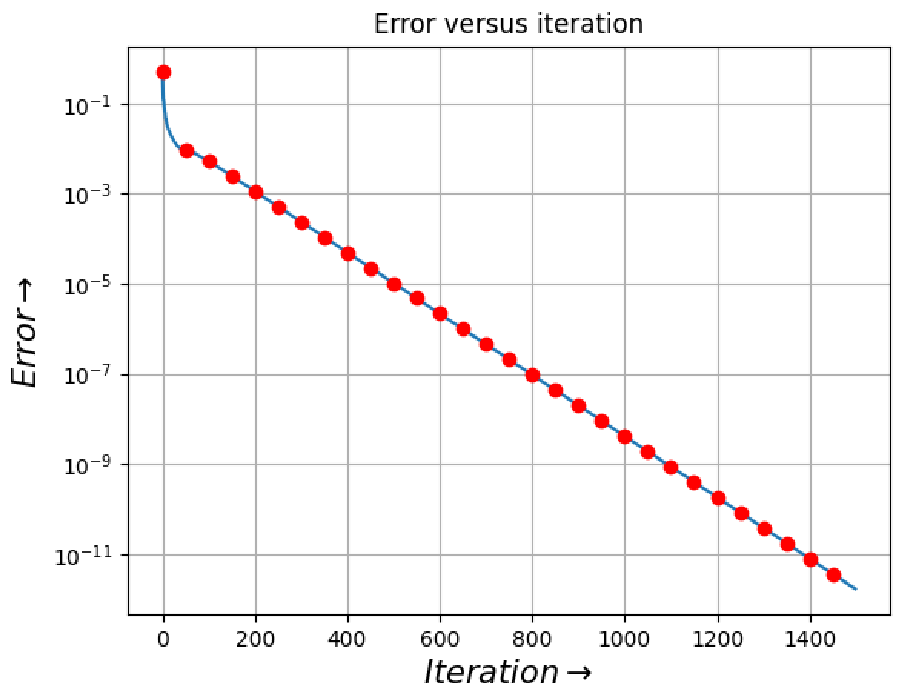
- The error calculated can be analysed by plotting it against the iterations
- Using semilogy and loglog plots, For better visualisation consider only 50th point.

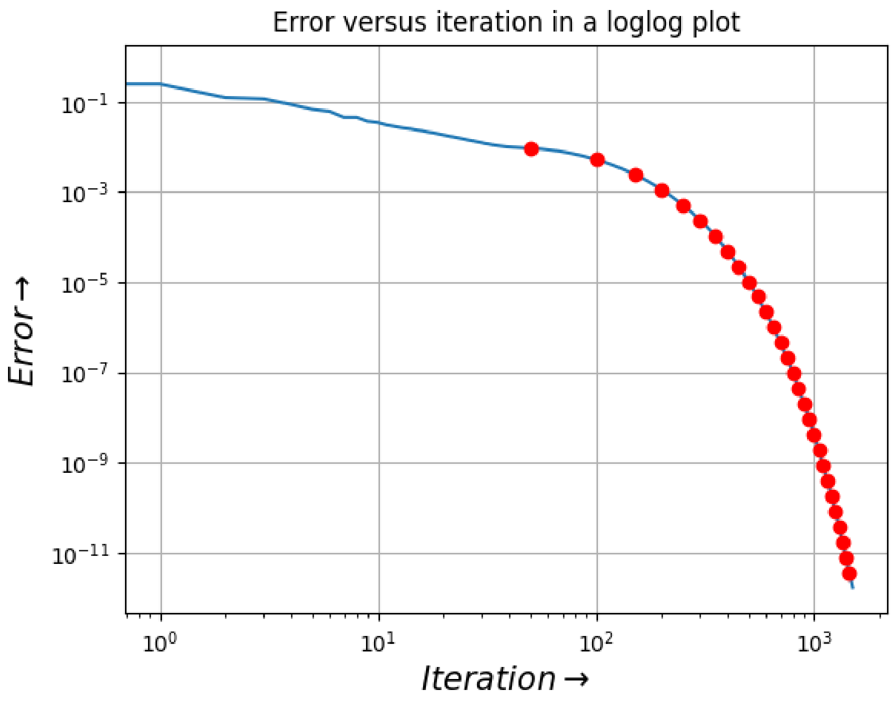
The python code snippet for plotting the graphs are as shown:

```
figure(1)
semilogy(n,errors)
semilogy(n[::50], errors[::50], 'ro')
title("Error versus iteration")
xlabel(r'$Iteration\rightarrow$', size = 15)
ylabel(r'$Error\rightarrow$', size = 15)
grid(True)

# Plotting of error vs iteration in loglog.
figure(2)
loglog(n,errors)
loglog(n[::50], errors[::50], 'ro')
title("Error versus iteration in a loglog plot")
xlabel(r'$Iteration\rightarrow$', size = 15)
ylabel(r'$Error\rightarrow$',size = 15)
grid(True)
```

The respective error plots are as shown:





4.1 Fitting the error

We note that the error is decaying exponentially for higher iterations.I have plotted 2 fits. One considering all the iterations(fit1) and another without considering the first 500 iterations. There is very little difference between the two fits that too before 500 iterations. After 500th iteration there is no difference, Can be seen through comparing original and fit2.

The python code is as follows:-

```
def Fitting_Exp(x, A, B): # Defining the function Fitting_Exp(x, A, B)
    return A*np.exp(B*x) # Returning the value to A*np.exp(B*x)

def Error_fitting(x, y): # Defining the function Error_Fitting(x, y)
    logy = np.log(y)

    vector_x = np.zeros((len(x), 2))
    vector_x[:, 0] = x
    vector_x[:, 1] = 1
    B, logA = np.linalg.lstsq(vector_x, np.transpose(logy))[0]
    return (np.exp(logA), B)

c_approx_500 = lstsq(c_[ones(Niter - 500), array(range(500, Niter))],
log(errors[500:]), rcond = None)
A_500, B_500 = exp(c_approx_500[0][0]), c_approx_500[0][1]
print("The values of A and B for the iterations after 500 are: ", A_500, B_500)

c_approx = lstsq(c_[ones(Niter), array(range(Niter))], log(errors), rcond = None)
A, B = exp(c_approx[0][0]), c_approx[0][1]
print("The values of A and B are: ", A, B)

fig3, ax1 = plt.subplots()
ax1.semilogy(range(Niter)[::50], errors[::50], label = 'original')
ax1.semilogy(range(Niter)[::50], Fitting_Exp(range(Niter)[::50], A, B), label = 'fi
ax1.semilogy(range(Niter)[::50], Fitting_Exp(range(Niter)[::50], A_500, B_500), lab
title("Best fit for error on semilog scale ")
xlabel(r'$Iteration\rightarrow$', size = 15)
ylabel(r'$Error\rightarrow$', size = 15)
grid(True)
legend()

fig4, ax2 = plt.subplots()
ax2.loglog(range(Niter)[::50], errors[::50], label = 'original')
ax2.loglog(range(Niter)[::50], Fitting_Exp(range(Niter)[::50], A, B), label = 'fit1
ax2.loglog(range(Niter)[::50], Fitting_Exp(range(Niter)[::50], A_500, B_500), label
title("Best fit for error on loglog scale ")
```

```
xlabel(r'$Iteration\rightarrow$', size = 15)
ylabel(r'$Error\rightarrow$', size = 15)
grid(True)
legend()
```

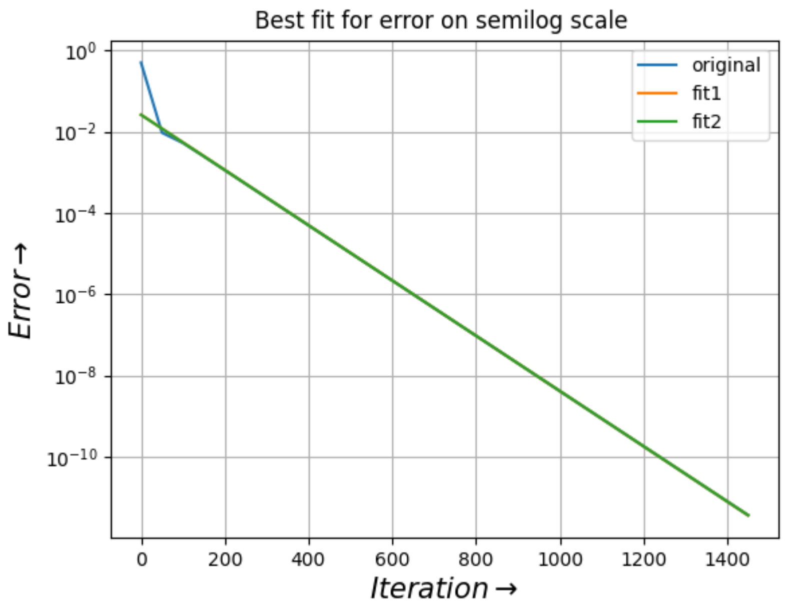


Figure 2: Best fit of error

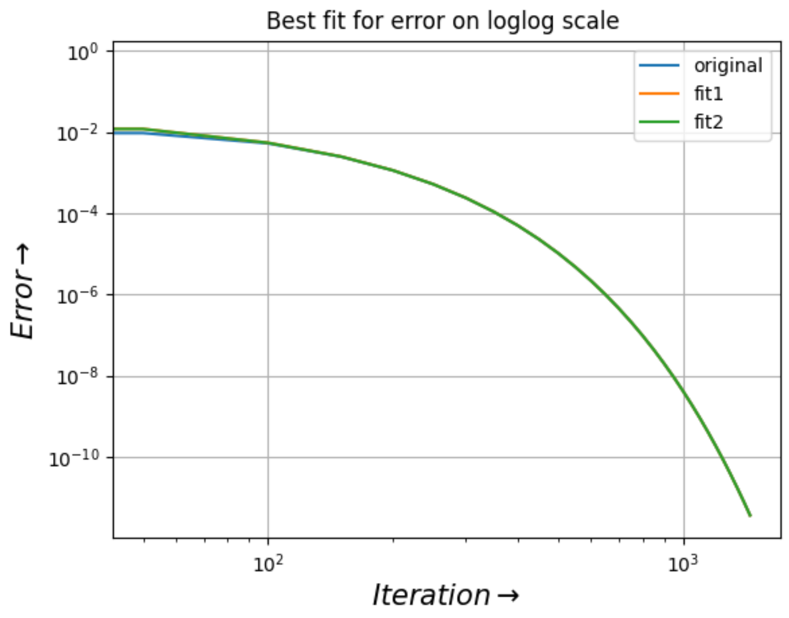


Figure 3: best fit of error

Upon execution of the code, we can get that

- $A_{500} = 0.02604399$ and $B_{500} = -0.01564807$
- $A = 0.02621557$ and $B = -0.01565526$

5 Surface Plot of Potential

We can analyse the potential variations by plotting it as a surface plot. The Python code is as follows:-

```
fig1=figure(6)
ax = p3.Axes3D(fig1, auto_add_to_figure = False)
fig1.add_axes(ax)
title("The 3-D surface plot of the potential")
xlabel(r'$X\rightarrow$')
ylabel(r'$Y\rightarrow$')
surf = ax.plot_surface(X, Y, phi, rstride = 1, cstride = 1, cmap = cm.jet)
fig1.colorbar(surf)
```

The surface plot is as shown below:

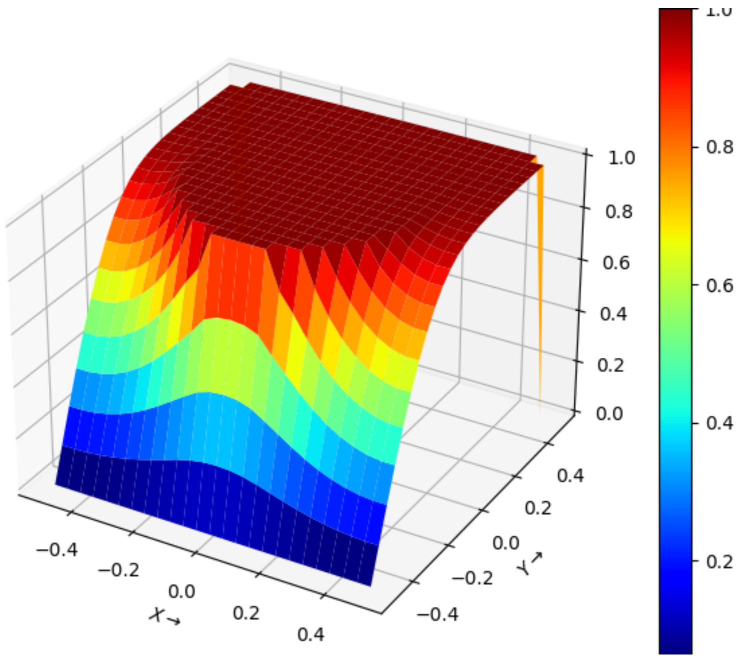


Figure 4: 3-D Surface potential plot of potential

6 Contour Plot of the Potential

We can analyse the potential variations by plotting it as a contour plot. The Python code is as follows:-

```
figure(5)
contourf(X,Y,phi)
plot(ii[0]/Nx - 0.48, ii[1]/Ny - 0.48, 'ro', label = "V = 1")
title("Contour plot of potential")
xlabel(r'$X\rightarrow$')
ylabel(r'$Y\rightarrow$')
colorbar()
grid(True)
legend()
```

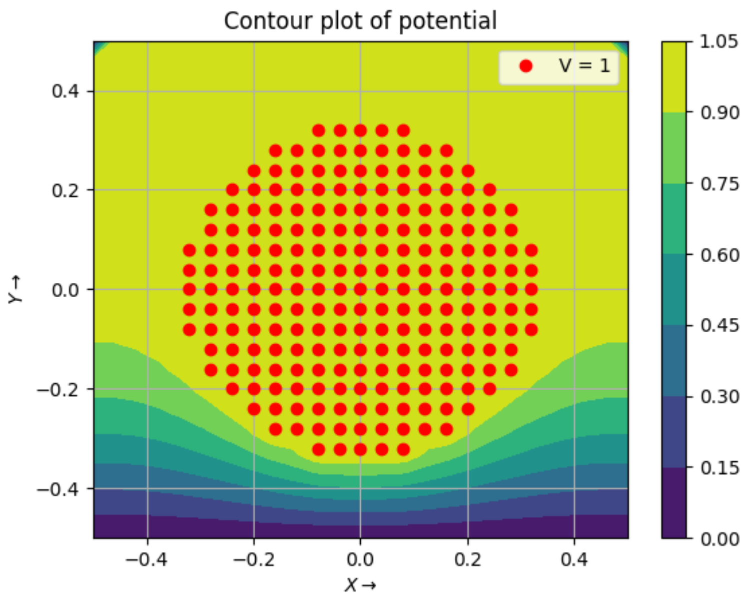


Figure 5: Contour plot of potential

7 Vector Plot of Currents

- The currents in the system in Cartesian form can be expressed as :

$$J_x = -\frac{\partial \phi}{\partial x} \tag{12}$$

$$J_y = -\frac{\partial \phi}{\partial y} \tag{13}$$

- Numerically, this can be expressed as

$$J_{x,ij} = \frac{1}{2}(\phi_{i,j-1} - \phi_{i,j+1}) \tag{14}$$

$$J_{y,ij} = \frac{1}{2}(\phi_{i-1,j} - \phi_{i+1,j}) \tag{15}$$

The python code for calculating the the current densities and plotting is as follows:

```
Jx = zeros((Ny, Nx)) # Initialisation
Jy = zeros((Ny, Nx))
Jx[1:-1, 1:-1] = 0.5*(phi[1:-1, 0:-2] - phi[1:-1, 2:])
```

```
Jy[1:-1, 1:-1] = 0.5*(phi[2:, 1:-1] - phi[0:-2, 1:-1])

# plotting of the current vector plot along with the potential.
figure(7)
quiver(X, Y, Jx, Jy)
plot(ii[0]/Nx - 0.48, ii[1]/Ny - 0.48, 'ro')
title("Vector plot of the current flow")
xlabel(r'$X\rightarrow$')
ylabel(r'$Y\rightarrow$')
show()
```

The vector plot of the current flow along with the potential is as shown below:

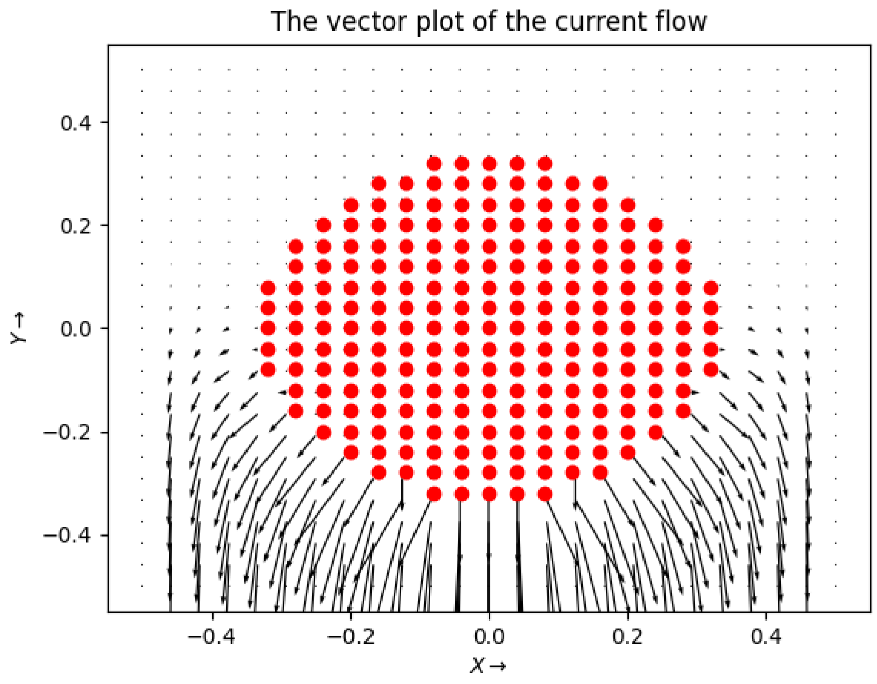


Figure 6: Vector plot of current flow

- So as we noted that the potential gradient was higher in down region of the plate, and we know that Electric field is the gradient of the potential as given below

$$\vec{E} = -\nabla\phi \tag{16}$$

- So $\vec{E}$ is larger where there is potential gradient is high and is inverted since it is negative of the gradient!, So it is higher in down region which is closer to bottom plate which is grounded
- And we know that

$$\vec{J} = \sigma\vec{E} \tag{17}$$

- So $\vec{J}$ is higher and perpendicular to equi-potential electrode region i.e "Red dotted region" so the current is larger in down part of the plate and perpendicular to the red dotted electrode region since $I = \vec{J}.\vec{A}$
- we notice that hardly any current flows through the top part of the wire. With a little thought, we observe that the lower surface being grounded, the easiest way for charge carriers to flow from the electrode would be directly through the lower half of the wire, thus avoiding a longer, more resistive path through the top half of the wire.

8 Conclusion :

- On analysing the quiver plot of the currents, it was noticed that the current was mostly restricted to the bottom of the wire, and was perpendicular to the surface of the electrode and the conductor.
- The currents in the plate are normal to the contour lines. In addi-tion, most of the current flow is between the two electrodes with somefringing of the current near the edges
- 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.
- Very little current flows through the other edges of the plate. This is primarily because most of the drop in the potential on the plate isb etween the central region and the top edge of the plate (as is clear from the surface plot).
- We observe that the best method to solve this is to increase N_x and N_y to very high values (100 or ≥ 100) and increase the number of iterations too, so that we get accurate answers i.e currents in the resistor.
- Using a finite differentiation approximation, we have found a solution toLaplace's equation for a given system. The error is seen to decay at ahighly gradual pace. Thus the chosen method of solving Laplace's equationis inefficient.