Assignment 5: The Laplace Equation

Bhargav Vimpadapu [EE20B151]

March 11, 2022

Abstract

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.

1   Introduction

This report will discuss about the solver for the currents in a resistor, the current's dependency on the shape of the resistor and also which part of the resistor is likely to get hottest. Here we analyse the currents in a square copper plate to which a wire is soldered to the middle of it. 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.

- 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 sides are floating. The plate is 1 cm by 1 cm in size.

2   Potential Array and Initialization

- 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 :

$$X^2 + Y^2 \leq 0.35^2 \tag{1}$$

- Then assign 1 V to those indices.

The python code snippet is as shown:

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

phi = zeros((Ny,Nx))
x = linspace(-0.5,0.5,Nx)
y = linspace(-0.5,0.5,Ny)
n = arange(Niter)
X,Y = meshgrid(x,-y)

A = (X*X) + (Y*Y)
ii = where(A <= (0.35*0.35))
phi[ii] = 1.0
figure(1)
plot(ii[0]/Nx-0.48,ii[1]/Ny-0.48,'ro',label="V = 1")
title("Initial Potential Contour")
xlim(-0.5,0.5)
ylim(-0.5,0.5)
xlabel(r'$X\rightarrow$')
ylabel(r'$Y\rightarrow$')
grid(True)
legend()
```

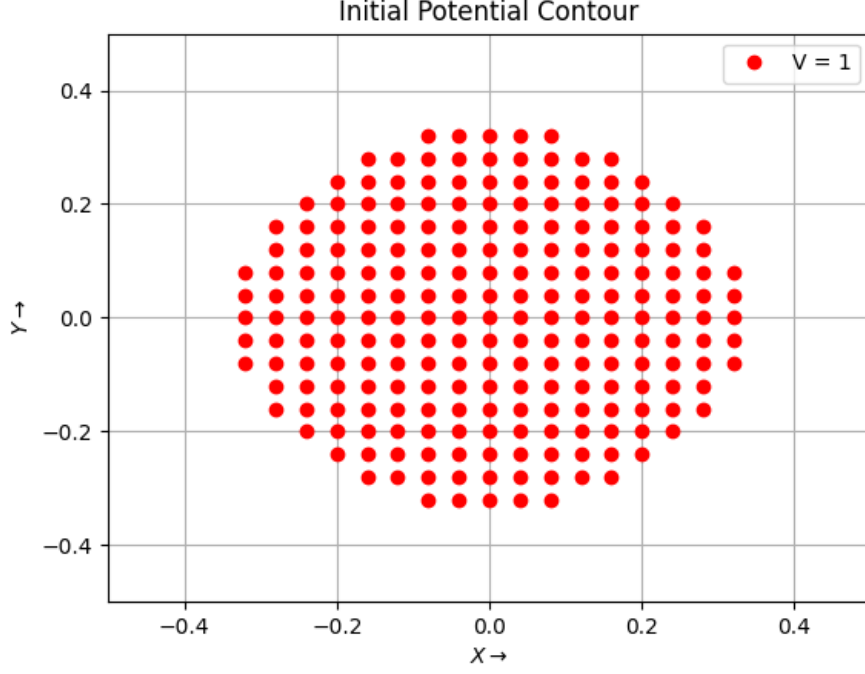The plot for the initial potential contour is as shown:

Figure 1: Contour plot of initial potential

3 Potential array updation and iteration

- 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 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

$$\frac{\partial \phi}{\partial n} = 0 \tag{3}$$

The python code snippet 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:,1:-1])
    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 Calculations

- The error calculated in the last section can be analysed by plotting it against the iteration number.

- This will give us a much more clearer picture of how the the error varies with respect to the iteration number.

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

```
# Plotting of error vs iteration in semilog.
figure(2)
semilogy(n,errors)
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(3)
loglog(n,errors)
title("Error versus iteration in a loglog plot")
xlabel(r'$Iteration\rightarrow$',size=15)
ylabel(r'$Error\rightarrow$',size=15)
grid(True)
# Plotting of error vs iteration above 500 in semilog .
figure(4)
semilogy(n[500:],errors[500:])
title("Error versus iteration above 500")
xlabel(r'$Iteration\rightarrow$',size=15)
ylabel(r'$Error\rightarrow$',size=15)
grid(True)
# Plotting of actual and expected error (above 500 iterations) in semilog.
figure(5)
semilogy(n[500:],errors[500:],label="Actual")
semilogy(n[500:],a_500*exp(b_500*n[500:]),label="Expected")
title("Expected versus actual error (>500 iterations)")
xlabel(r'$Iteration\rightarrow$',size=15)
ylabel(r'$Error\rightarrow$',size=15)
grid(True)
legend()
```

The respective error plots are as shown:

Error vs Iteration above 500
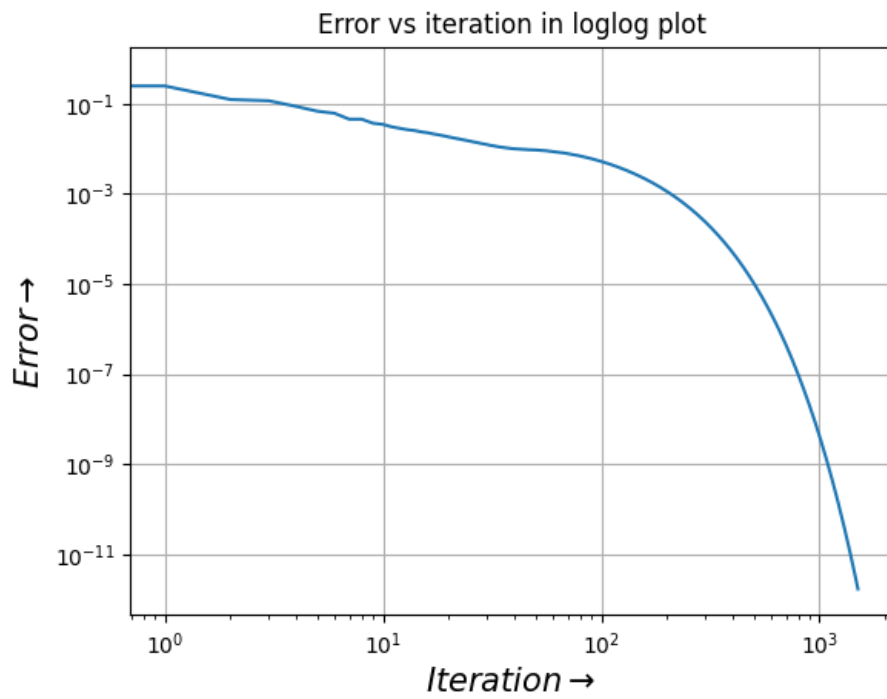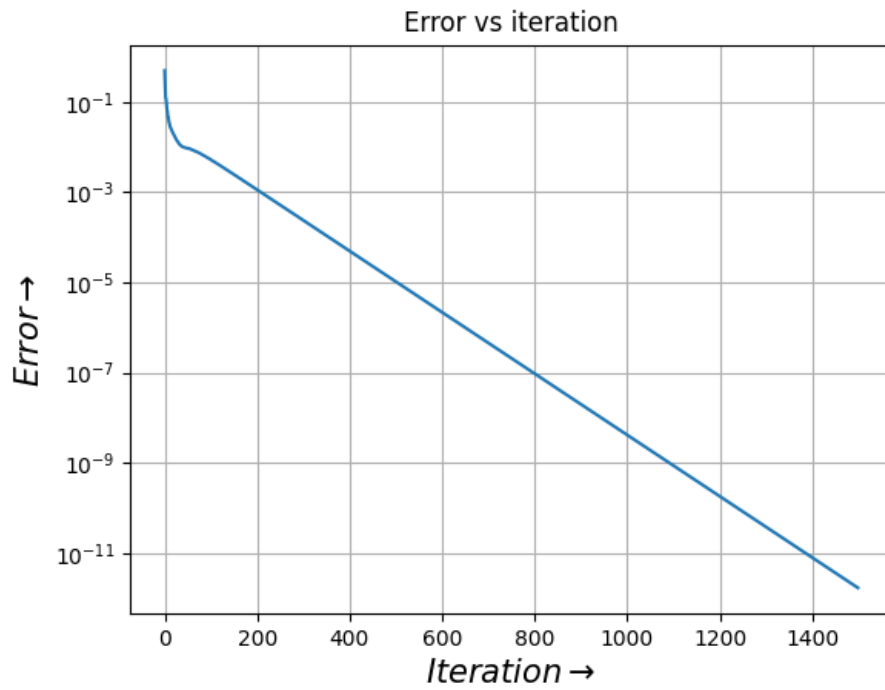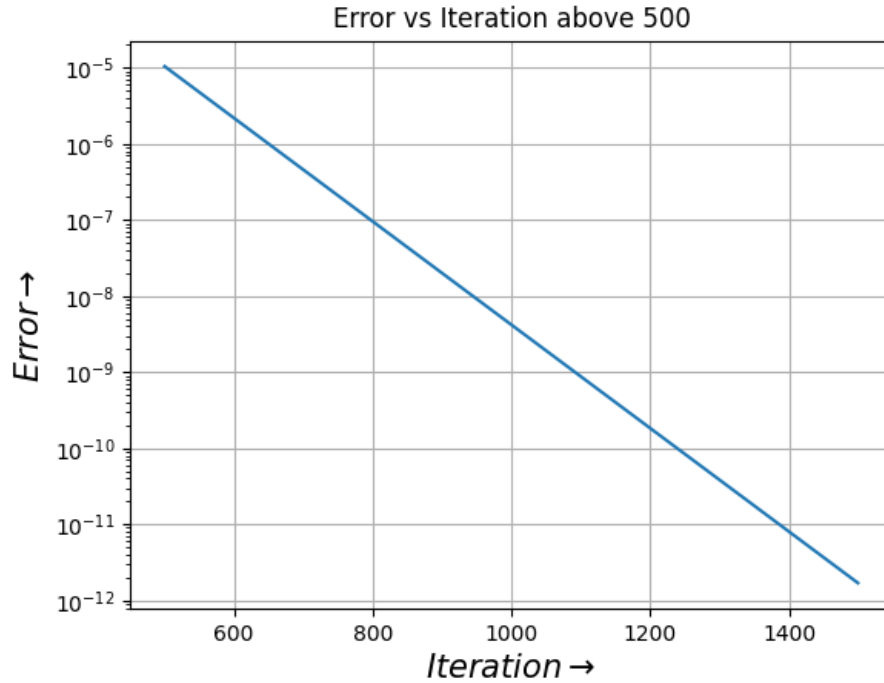
Extracting the exponential

- We can find the fit using Least squares for all iterations 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.

$$log y = log A + Bx \tag{4}$$

- We can find the constants of the error function obtained for the two cases using lstsq and compare them.

The python code snippet to extract the exponent is as follows:

```
c_approx_500 =
lstsq(c_[ones(Niter-500),arange(500,Niter)],log(errors[500:]),rcond=None)
a_500,b_500 = exp(c_approx_500[0][0]),c_approx_500[0][1]
c_approx = lstsq(c_[ones(Niter),arange(Niter)],log(errors),rcond=None)
a, b = exp(c_approx[0][0]), c_approx[0][1]
```

The plots comparing both the errors are as shown:

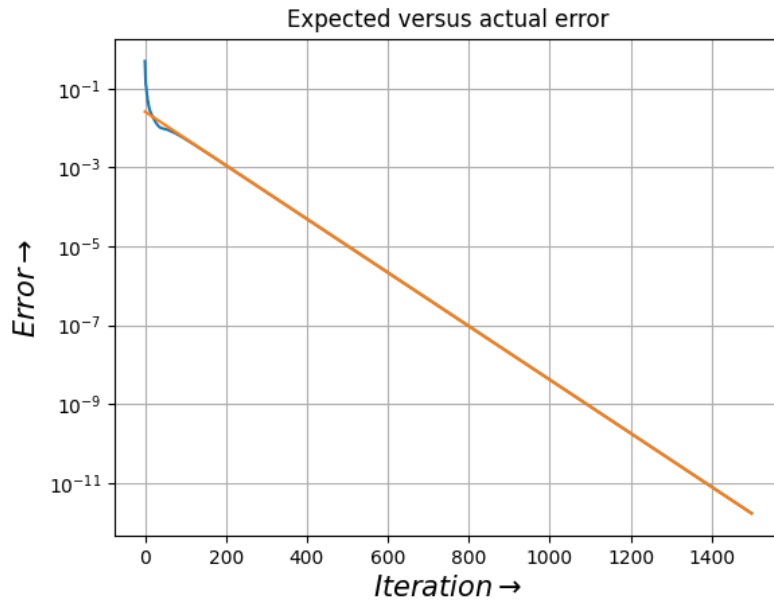Figure 2: Expected versus actual error (above 500 iterations)

Figure 3: Expected versus actual error

Upon execution of the code, we can observe that

- $A_{500} = 0.02604399$ and $B_{500} = -0.01564807$

- $A = 0.02621557$ and $B = -0.01565526$

5 Surface and Contour Plot of Potential

We can analyse the potential variations by plotting it as a surface and contour plot. The following python code explains it.

```
figure(7)
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()

fig1=figure(8)
ax=p3.Axes3D(fig1)
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)

show()
```
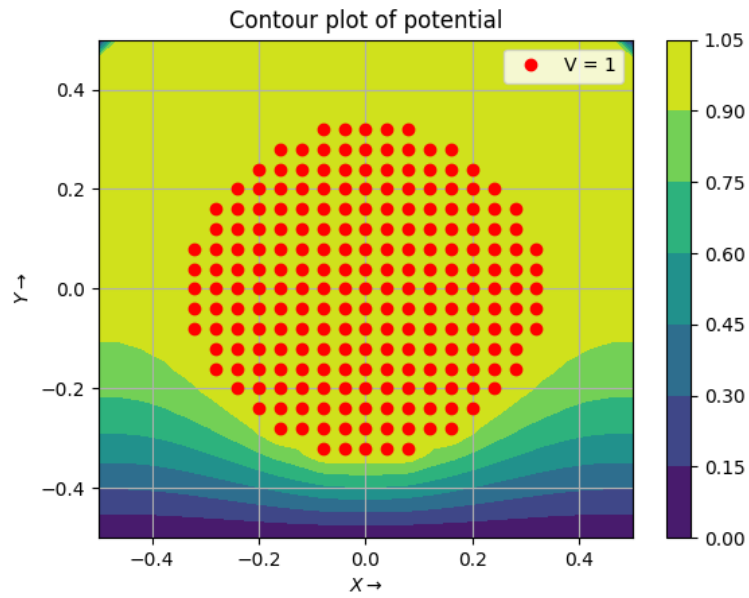
The surface and contour plots are as shown below:
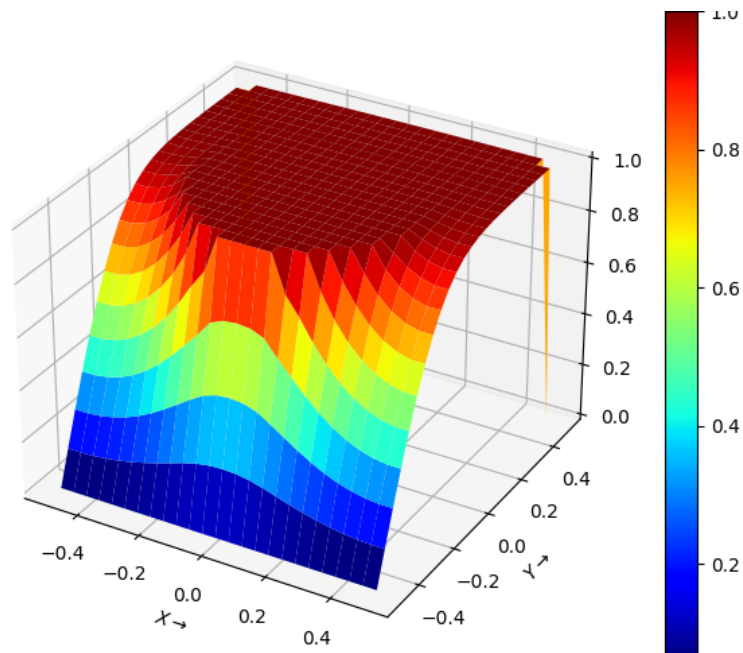

Figure 4: Contour plot of potential

Figure 5: 3-D Surface potential plot of potential

6 Vector Plot of Currents

- We can 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

$$J_x = -\frac{\partial \phi}{\partial x} \tag{5}$$

$$J_y = -\frac{\partial \phi}{\partial y} \tag{6}$$

- To program this we use these equations as follows:

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

$$J_{y,ij} = \frac{1}{2}(\phi_{i-1,j} - \phi_{i+1,j}) \tag{8}$$

The python code snippet for calculating the the current densities and plotting them are as follows:

```
Jx = zeros((Ny, Nx))
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])

figure(0)
quiver(X,Y,Jx,Jy)
plot(ii[0]/Nx-0.48,ii[1]/Ny-0.48,'ro')
title("The vector plot of the current flow")
xlabel(r'$X\rightarrow$')
ylabel(r'$Y\rightarrow$')
```

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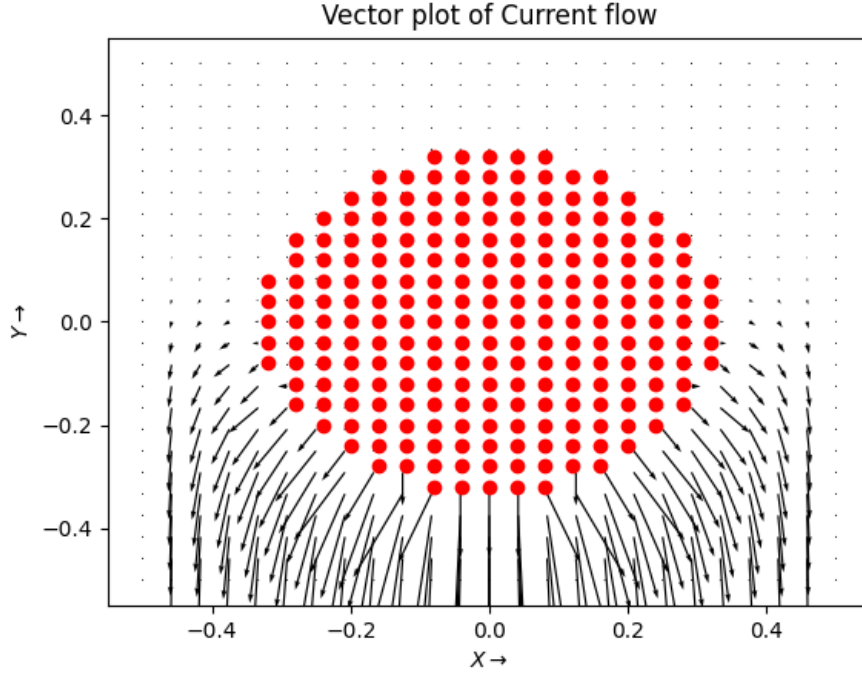

Figure 6: Vector plot of current flow

- 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

$$\vec{E} = -\nabla \phi \qquad (9)$$

- 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

$$\vec{J} = \sigma \vec{E} \qquad (10)$$

- So $\vec{J}$ is higher and perpendicular to equipotential 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 $I = \vec{J}.\vec{A}$

12

- This is the reason most of the current flows from electrode to the bottom plate which is grounded because of higher potential gradient.

- And there is almost zero current in upper part of the plate since there is not much potential gradient as we observed from the surface and contour plot of the potential ϕ

7   Conclusion :

- To conclude , most of the current is in the narrow region at the bottom. So this region 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 ≥ 100) and increase the number of iterations too, so that we get accurate answers i.e currents in the resistor.

- But the trade off 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.