EE2703 : Applied Programming Lab
Assignment - 5

Jawhar S
EE20B049

March 11, 2022

Overview of Solution to the Resistor Problem

In this assignment, the Laplace Equation for electric fields is solved using an iterative method in Python.

To solve for the potential function of a circular voltage probe on a square resistive sheet, firstly the objects referring to the physical elements and dimensions were initialized. The potential function was initialized with appropriate boundary conditions. Then, a formulation of the differential equation in the array index space allowed for estimating the potential using the cell's neighbours. The update rule was applied over many iterations to get the potential. The potential was also visualized on surface and contour plots. The current was then calculated and plotted in vector form. It was observed that most of the current flows through the bottom half of the plate.

The detailed process and observations from each of the plots is given below.

Numerical Solution to the Laplace Equation

Laplace's equation in 2-dimension can be written in Cartesian coordinates as

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

An approximate solution for the above equation for a 2-dimensional grid of points is

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

The potential at any point is the average of potential values at its neighbouring points. At each point, we replace the potential by the average of its neighbours. We continue to iterate till the solution converges (i.e. The maximum change in elements of ϕ is less than some tolerance).

At boundaries where the electrode is present, we don't change the value of potential. At boundaries where there is no electrode, the current should be tangential because charge can't leap out of the material into air. Since current is proportional to the Electric Field, the gradient of ϕ should be tangential.

The code for updating the potential at each iteration, applying boundary conditions is given below:

```
oldphi = phi.copy()
errors = np.zeros(Niter)
# Performing iteration
for k in range(Niter):
    oldphi = phi.copy()
    # Updating potential array
    phi[1:-1, 1:-1] = 0.25*(oldphi[1:-1, 0:-2] + oldphi[1:-1, 2:]
                        + oldphi[0:-2, 1:-1] + oldphi[2:, 1:-1])
    # Boundary conditions
    phi[1:-1, 0] = phi[1:-1,1]          # Left
    phi[1:-1, -1] = phi[1:-1, -2]       # Right
    phi[0, 1:-1] = phi[1, 1:-1]         # Top
    phi[-1, 1:-1] = 0                   # Bottom (Ground)
    # Re-Updating the wire potential
    phi[ii] = 1.0
```

Initial Setup

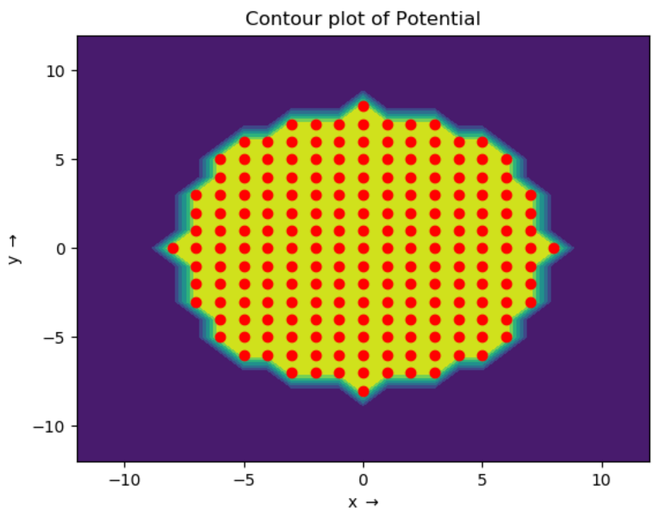


Figure 1: Contour plot of Potential

In the above contour plot, the red dots represents the points at which $\phi = 1$.

The parameters N_x, N_y, Radius and Number of Iterations are initialized as 25, 25, 8 and 1500 respectively. These parameters can be changed by the user via command-line arguments.

Error in iterative updation

The errors were plotted on semilog and loglog scales. The semilog plot showed a straight line, indicating an exponential behaviour of error.

The errors were fitted using two exponential functions, one using all the values and the other ignoring the first 500 values which did not match with the straight line semi-log plot.

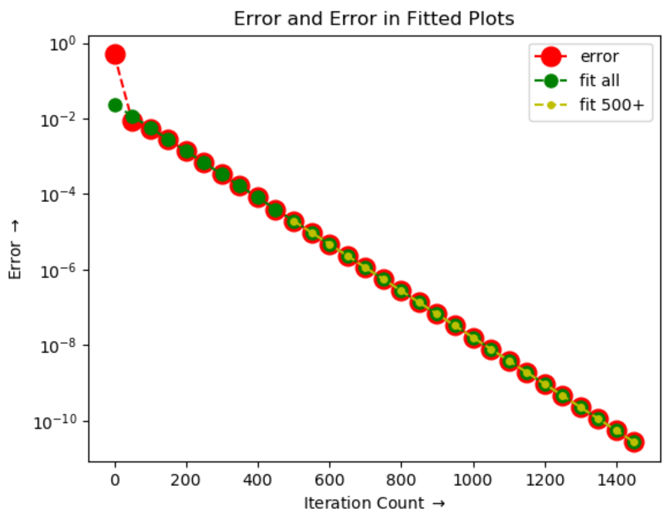


Figure 2: semi-log plot

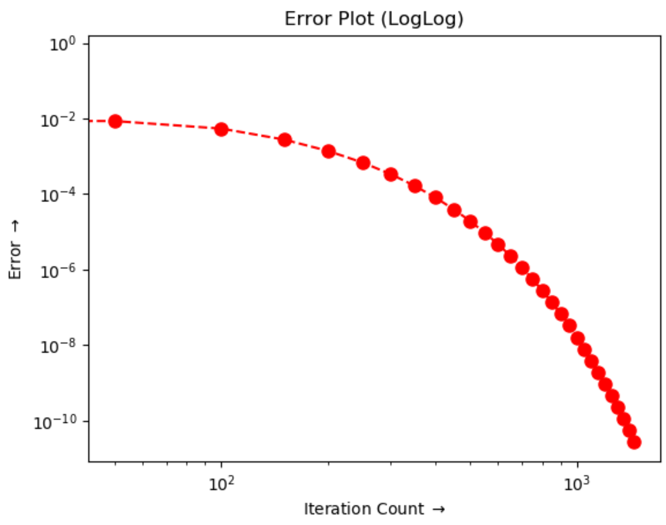


Figure 3: log-log plot

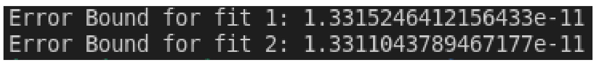
The code used for fitting the errors using two different set of values is given below

```
# Function to find ideal coefficients A, B for the fit y = A*exp(Bx)
def lstsq_fit(x,y):

    x = x.reshape((-1,1))
    one_arr = np.ones((x.shape[0],1))
    xvar = concatenate((one_arr,x), axis=1)
    logA, B = lstsq(xvar, log(y), rcond=None)[0]
    return exp(logA), B
```

Stopping condition

We find a bound on the error using the formula $\frac{A}{B}exp(B(N + 0.5))$. The function is exponential in Niter, implying that the error decreases exponentially with number of iterations. Further, instead of a fixed number of iterations, this error bound can be used as the stopping condition.



```
Error Bound for fit 1: 1.3315246412156433e-11
Error Bound for fit 2: 1.3311043789467177e-11
```

Figure 4: Maximum estimated error

Visualizing the Potential

The estimated potential is sketched using surface and contour plots. The potential is highest at the wire and the area above it, while it decreases towards the bottom.

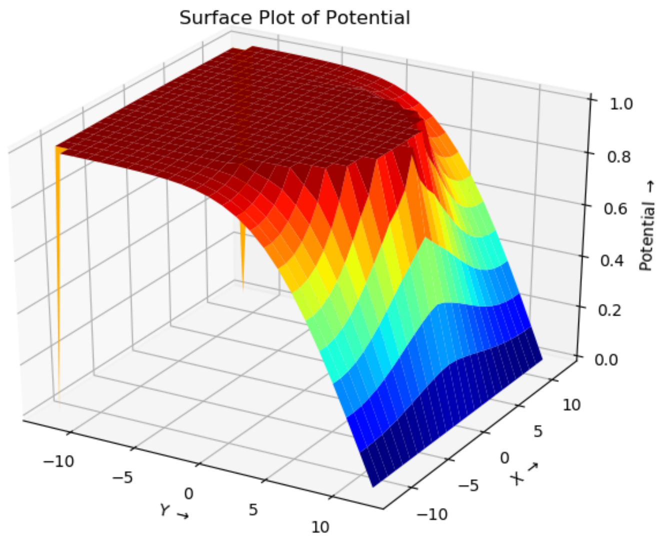


Figure 5: Surface Plot

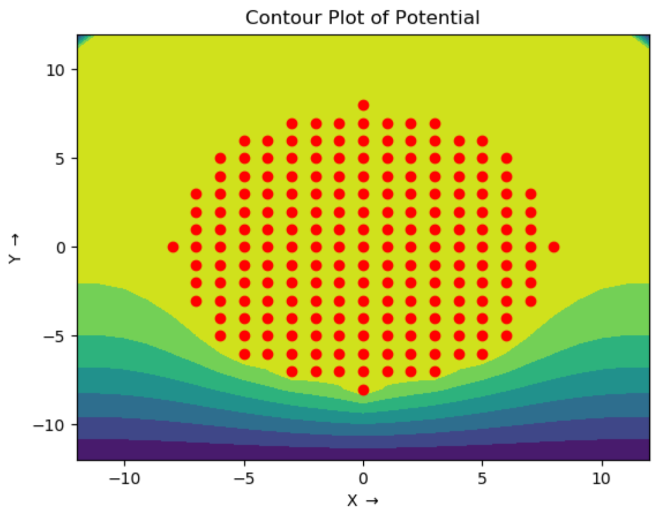


Figure 6: Contour Plot

Visualizing the Current Density

The current is found using the equations

$$J_x = -\sigma \frac{\partial \phi}{\partial x} J_y = -\sigma \frac{\partial \phi}{\partial y}$$

The code used for obtaining the current density at the surface of the copper plate is given below.

```
# Obtaining Current density at the surface
Jx = np.zeros(phi.shape)
Jy = np.zeros(phi.shape)
Jx[:,1:-1] = (phi[:,0:-2]-phi[:,2:])/2
Jy[1:-1,:] = (phi[2:, :]-phi[0:-2,:])/2
```

Taking $\sigma = 1$ and transforming the equation to index space, we sketch a quiver plot.

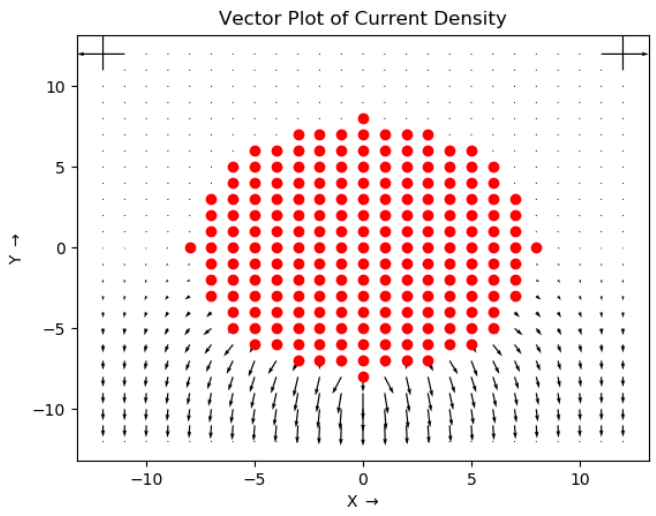


Figure 7: Quiver Plot

Most of the current exists in the bottom half of the plate. The current is nearly zero in the top half of the plate. This can be explained using equivalent resistance – paths with more resistance contain lesser current than paths parallel to it with lower resistance. Since the ground is at the bottom of the plate, the shortest and therefore the least resistance path is to move downwards.

Conclusion

This assignment explored the iterative method of solving the Laplace Equation using discrete differentiation. The potential and current density plots for the given configuration were obtained. It is noticed that the principle of least equivalent resistance intuitively explains the current distribution.

Through this assignment, the process of using vectorization in Numpy became clear. A variety of plots like semi-log, log-log, 3D surface, contours and quiver plots were used to effectively visualize the key ideas.