Assignment No 5

Subhankar Chakraborty

March 7, 2019

1 Introduction

This assignment deals with finding currents in a resistor. The currents depend on the shape and size of the resistor as well as the boundary conditions.

1.1 Laplace's equation

In a given conductor let there be a *current density* **J** at a given point. The current density is related to the *Electric field* **E** at that point by the equation

$$\boxed{\mathbf{J} = \sigma \mathbf{E}}$$

where σ is the conductivity of the material, which is a constant for a given material at a given temperature. Now the Electric field **E** is the negative gradient of the potential ϕ .

$$\boxed{\mathbf{E} = -\nabla \phi}$$

From the continuity equation we get

$$\boxed{\nabla . \mathbf{J} = -\frac{\partial \rho}{\partial t}}$$

where ρ is the change density at that point. Combining these equations we get

$$\nabla .(-\sigma \nabla \phi) = -\frac{\partial \rho}{\partial t}$$

Assuming a constant value of conductivity(σ) which is true for a lot of real life conductors, the equation becomes

$$\boxed{\nabla^2 \phi = \frac{1}{\sigma}\frac{\partial \rho}{\partial t}}$$

This is the **Laplace Equation**.

1.2 The DC Current Case

For DC currents, there is no change accumulation in any part of the circuit at any point of time. So the value of the charge density(ρ) at a point does not change with time. This makes the term $\frac{\partial \rho}{\partial t}$ zero. Thus the Laplace Equation for DC currents becomes

$$\boxed{\nabla^2 \phi = 0}$$

2 Numerical Solution in 2-Dimensions

Writing the derivatives in a two dimensional plane we get

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

After a few numerical simplifications, this equation in a plane with finite precision becomes

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

ie, the potential at any point is the average of the potentials of its surrounding points. Repeating this over many iterations while keeping in mind the boundary conditions, the potential at each point converges to some value. At boundaries where electrodes are present, the potential takes the value of the electrode potential. At boundaries where no electrodes are present, current should be tangential to the surface as change cannot leap out of material into thin air. Since the current density (and hence, current) is proportional to the electric field, this boundary condition ensures that the **gradient of ϕ in the normal direction to the boundary is zero.**

3 Coding the Potential

3.1 Initialization

We first initialize a matrix of size (N_x, N_y) with all entries initialized to zero. This is the value to which ϕ is initialized.. The following section of code does it.

```
X,Y = np.meshgrid(x,y)
potential  = np.zeros((Nx+1, Ny+1))
```

Now we adjust the coordinates such that we get a region of dimensions $1cm * 1cm$. Also we need to ensure that $y = 0$ corresponds to the centre of the plate (ground) and not the top as how a programming language will interpret it. This can be done as below.

```
Y = Y[::-1]
X = (X-Nx/2)/Nx
Y = (Y-Ny/2)/Ny
```

After doing the above we find the region in which the electrode lies and set its potential to 1. This can be done by setting the coordinates of the origin to centre of the matrix(already done in the previous step) and finding the region which lies within a distance of the radius from the origin.

```
potential[X*X+Y*Y<=radius*radius]= 1.0
```

Figure 1 will give a better idea as to our setup.

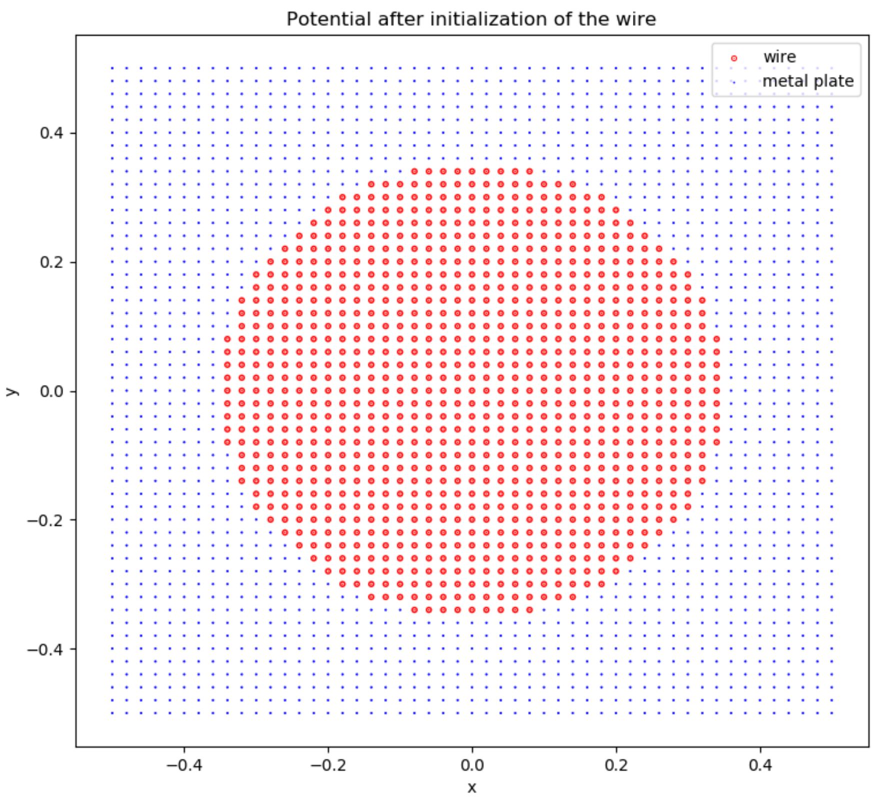

Figure 1: Potential after initialization of the electrode.

3.2 Performing the Iterations

During the iterations we keep a variable named *oldpotential* to find how much is the error changing on each iteration. However setting *old potential*

= *potential* will be disastrous as this will simply equate their pointers and any change made in *potential* will be reflected in *old potential*, giving a consistent error value of zero. So we do this.

```
old_potential = potential.copy()
```

This ensures *old potential* takes only a copy of the value in potential and not the pointer itself, allowing us to measure the error.Ignoring boundary conditions as of now for the iteration step and using the equation we found for $\phi_{i,j}$,the update equation becomes

```
potential[1:-1, 1 :-1] = 0.25*(potential[1:-1, 2:]+
potential[1:-1,0:-2]+potential[2:, 1:-1]
+potential[0:-2, 1:-1])
```

This is performed for *Niter* number of times while keeping in mind the boundary conditions, which we will see in the next subsection.

3.3 Boundary Conditions

We need to keep three boundary conditions in mind.

- The potential of the plate which is in contact with the ground must be zero. This is not much of a concern as the lowest row is not a part of our update equation.
- The potential of the electrode must remain as it was initialized.This can be done exactly how we initialized the electrode.The following piece of code takes care of this.

  ```
  potential[X*X+Y*Y<=radius*radius]= 1.0
  ```

- **E** should be tangential to the other three boundaries. This means the gradient of ϕ along the normal direction should be zero. For example, $\frac{\partial \phi}{\partial y}$ must be zero in the top surface. This condition can be taken care of by the following piece of code.

  ```
  potential[0,1:-1] = potential[1,1:-1]
  ```

  The code for the other 2 cases are exactly similar.

Having taken care of the boundary conditions, we can calculate the error at the end of each iteration.

```
error[i] = np.max(abs(potential-old_potential))
```

4 Plotting the Graphs

4.1 Decreasing Error per Iteration

We observe the following from the error values as a function of number of iterations.

- The *loglog* plot of error versus number of iterations will be roughly linear upto 500 iterations and will take a steep slope after that.
- The *semilogy* plot of error versus number of iterations takes a sharp dip and then is almost linear for 500 and above iterations.

Assuming the error (y) is of the form

$$y = Ae^{Bx}$$

we take log on both sides.

$$log(y) = log(A) + Bx$$

where x is the number of iterations. We take 2 least fit curves, one for all x and another for x above 500. We plot it along with the original curves to see how close is the *semilogy* plot to a linear curve. The Figures 2,3 and 4 show the plots.

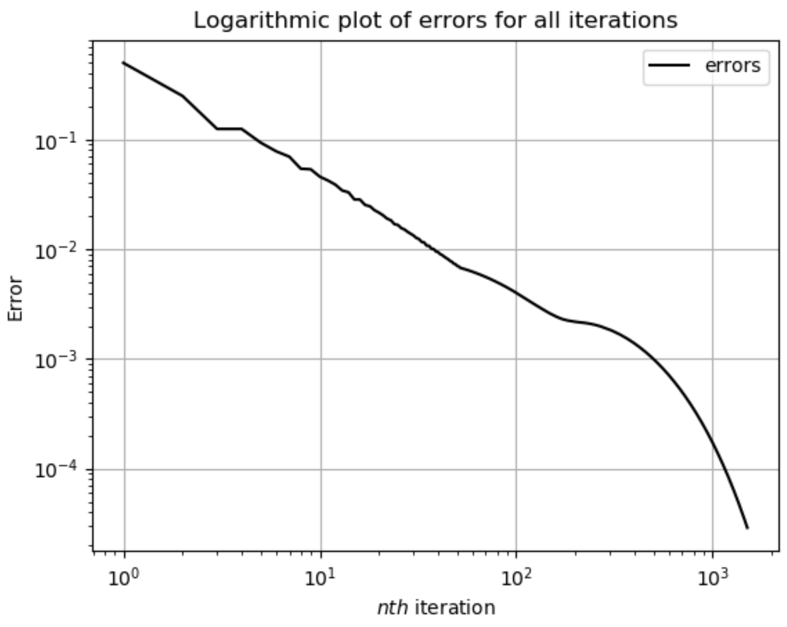

Figure 2: *loglog* plot of errors versus x

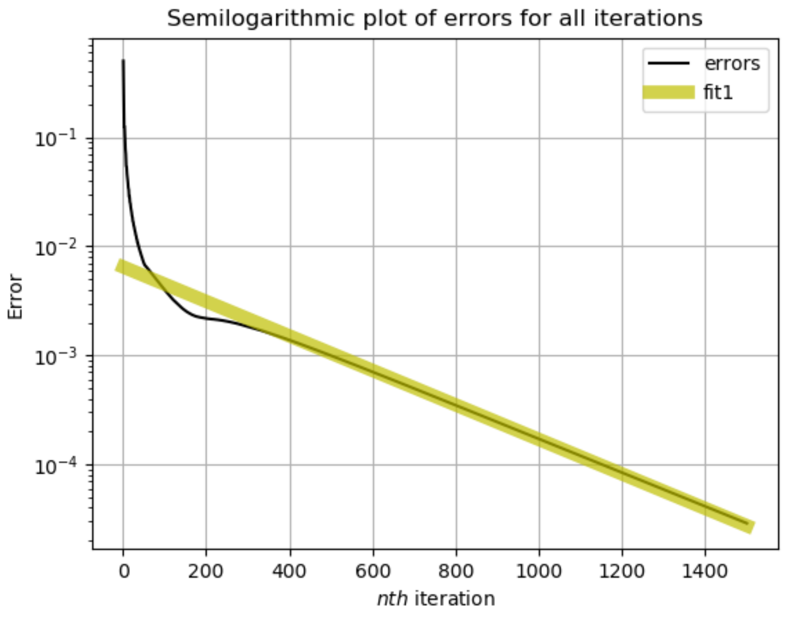

Figure 3: *semilogy* plot of errors versus all x

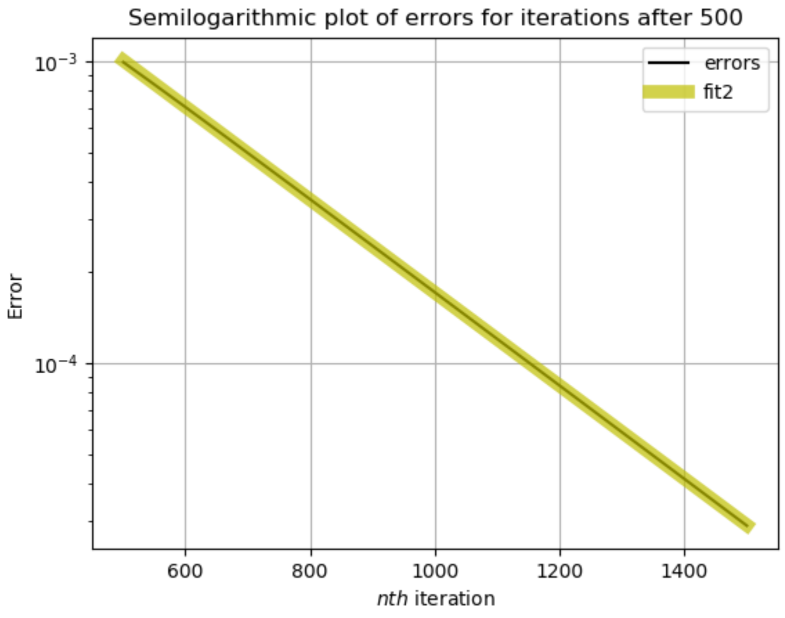

Figure 4: *semilogy* plot of errors versus x ($\forall x > 500$)

4.2 3D plotting ϕ

We can do a 3D plot of the potential obtained using the *Axes3D* module from matplotlib. The following code does it.

```
fig = plt.figure()
ax = fig.gca(projection = '3d')
ax.plot_surface(X,Y,potential)
plt.show()
```

The plot obtained is shows in figure 5.

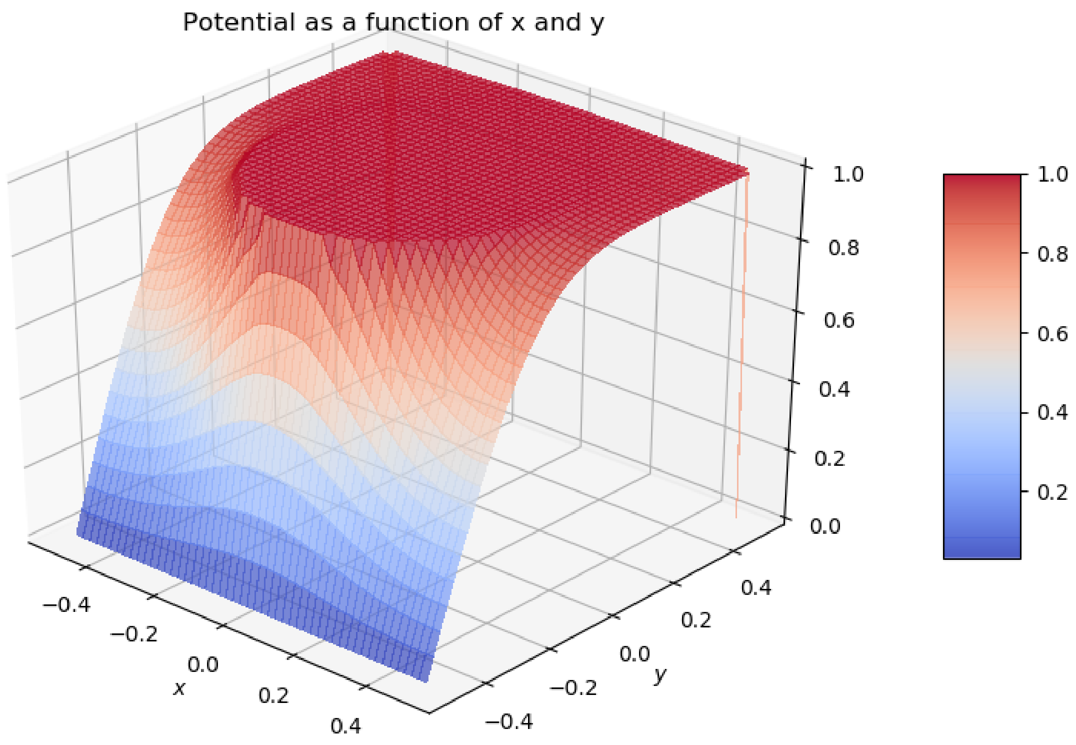

Figure 5: 3D plot of potential versus x and y

4.3 Contour Plot of ϕ

The contour plot of the potential along with the electrode is shown in Figure 6.

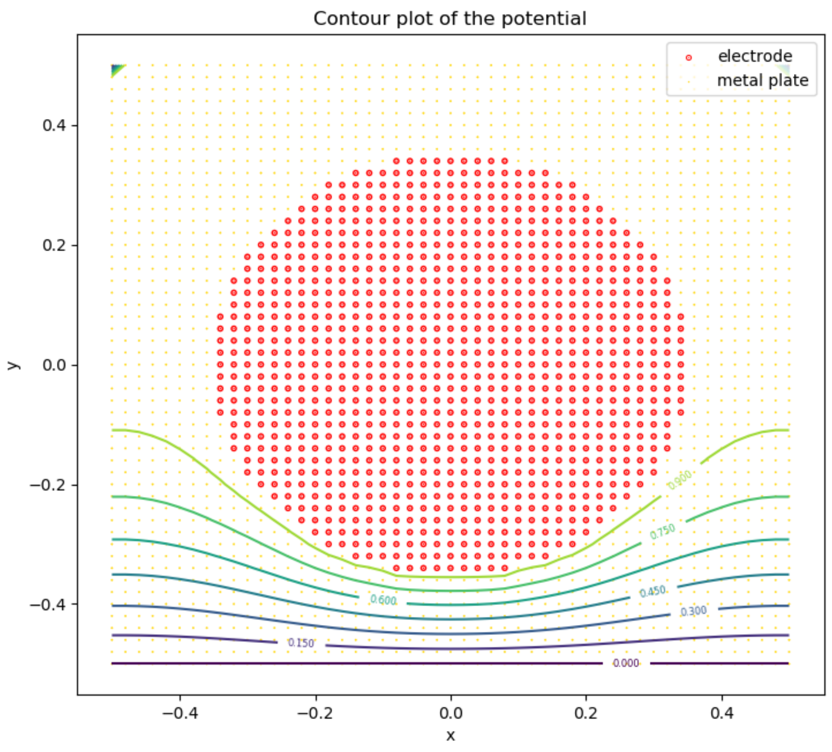

Figure 6: contour plot of potential

4.4 Vector plot of Current Densities

Since we are only plotting the relative magnitude of currents here, the value of σ does not make a difference here and so we can set it to unity. The equations we have now are then

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

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

where J_x and J_y are magnitudes of current densities in x and y directions respectively. This numerically translates to

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

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

This can be coded quite easily.

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

The quiver command is used to plot the directions of **J**. The code is

```
plt.quiver(X,Y,Jx,Jy,scale = 4e0)
```

The scale is added to make the arrows look proportionately sized. The plot can be seen in Figure 7.

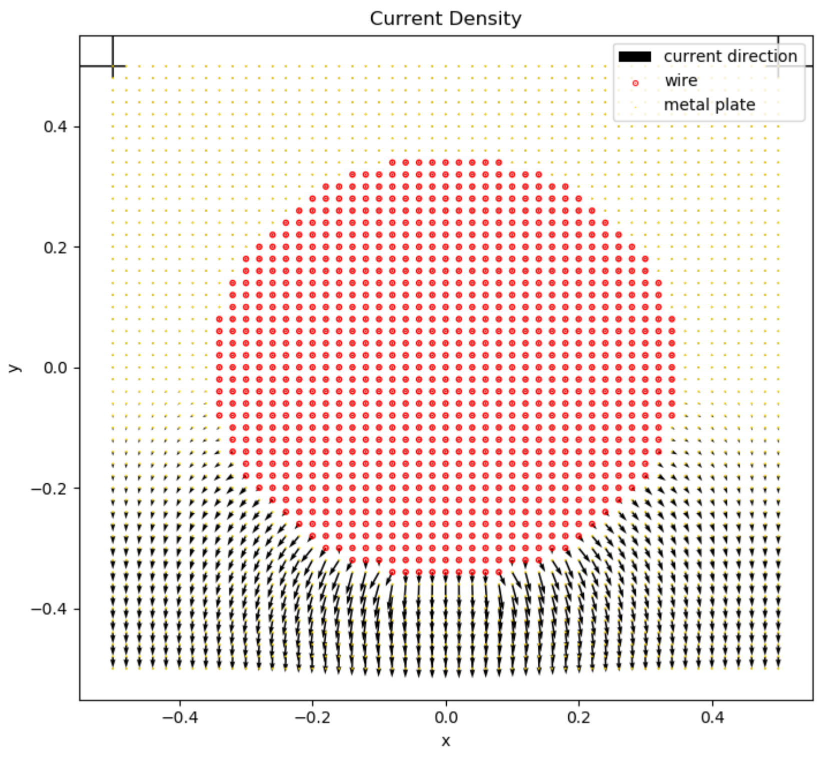

Figure 7: Current Density

5 Inference

One of the inferences that can be drawn from the current density is that almost no current flows from the top surface of the conductor. This should be obvious as the path between the ground and the lowest part of the electrode is the electrically shortest path and hence, most of the current is concentrated in that region. As we keep moving upwards the electrical path length keeps increasing and hence the current value keeps decreasing. This becomes almost zero beyond a point.

Extras

One point I observed is that the part where we equate the potential of a point to be the average of its four neighbouring points can be seen as the convolution of an averaging filter with the ϕ matrix with padding mode set to same. The averaging filter is shown below.

$$\begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 0 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix}$$

This can be achieved by properly defining this matrix and making a small change to the ϕ update equations.

```
#change1
AVERAGE_FILTER = np.array([[0,0.25,0],[0.25,0,0.25],[0,0.25,0]])

#change2
potential= convolve2d(potential, AVERAGE_FILTER, mode='same')
```

We are using the **convolve2d** module from scipy to perform this convolution.The results are exactly the same.

NOTE: Throughout this report vectors have been represented in bold rather than with the arrow on top. So $\mathbf{J}$ is a vector whereas J_x is a scalar.