EE2703 Assignment 5

Allu Yaswanth , EE20B007

March 12, 2022

Abstract

- To solve for potential and currents in a system.
- To solve 2-D Laplace equations in an iterative manner.
- To plot graphs to understand the 2-D Laplace equation.

1 Introduction

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

Ohm's law

$$\vec{J} = \sigma \vec{E} \tag{1}$$

Charge Continuity equation.

$$\nabla . \vec{J} = -\frac{\partial \rho}{\partial t} \tag{2}$$

From the above equations,

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

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

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

2 Tasks

2.1 Parameters

They are assigned default values, which will be corrected via commandline arguments

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

#For the case where parameters given through command line.
if(len(sys.argv)>1 and len(sys.argv)< 5 ):
    Nx = int(sys.argv[1])
    Ny = int(sys.argv[2])
    Niter = int(sys.argv[3])
elif(len(sys.argv)>= 5):
    print('No. of argumnets must be less than 5')
    exit()
```

2.2 Variable initialization

Create a zero 2-D array of size Nx x Ny and assign 1 to cordinates within radius 1 from center

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

2.3 Allocating potential and plotting it

```
#Points inside the circle
A = (X*X) + (Y*Y)
ii = where(A <= (0.35*0.35))

phi[ii] = 1.0 #phi = 1 for all the points inside the circle
```

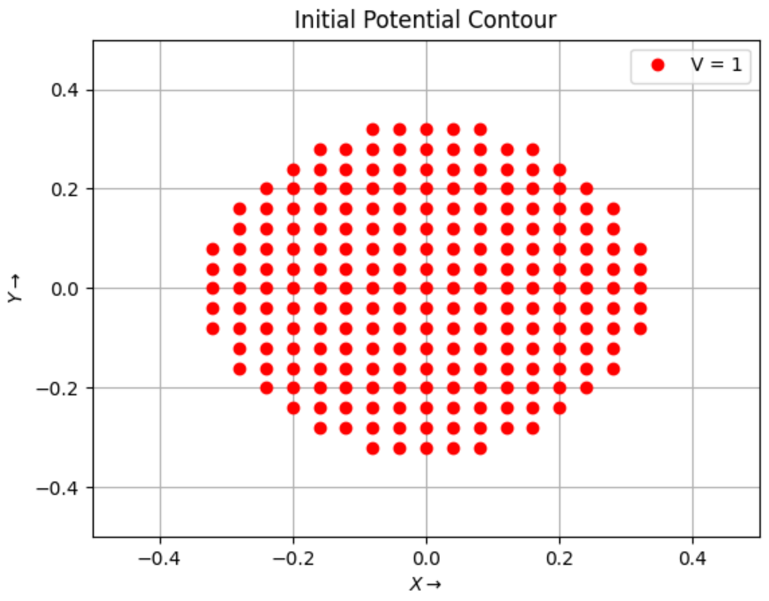

Figure 1: semilogy scale

2.4 Updating Potential

After converting the equation(4) to discrete domain, we can update matrix through iterations

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{4} \tag{5}$$

The bottom boundary is grounded. The other 3 boundaries have a normal potential of zero

```
#Perform the iteration and to calculate the error in the potential.
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])

#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()
```

2.5 Plotting the errors

We will plot the errors on semi-log and log-log plots. We can observe the error decreases very slow

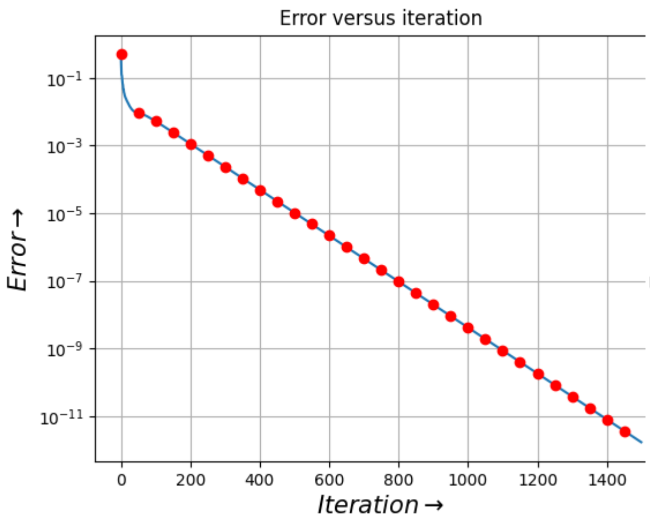

Figure 2: Error (semilog)

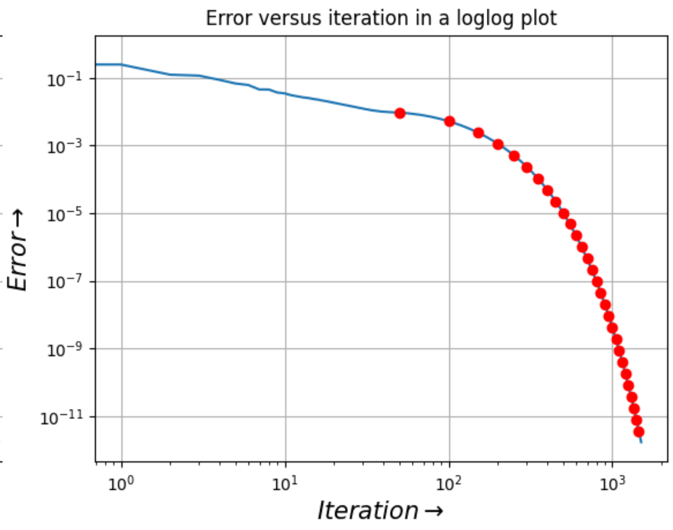

Figure 3: Error (loglog)

2.5.1 Fitting exponential curve

We observed that the error is decaying exponentially for higher iterations. We will fit two curves.

- Considering all iteration
- Considering after 500th iteration

```
# The exponent part of the error values.
c_approx_500 = lstsq(c_[ones(Niter-500),arange(500,Niter)],log(errors[500:]),rcond=None)#e
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),arange(Niter)],log(errors),rcond=None) #estimating laplace
A, B = exp(c_approx[0][0]), c_approx[0][1]
print("The values of A and B are: ",A,B)
```

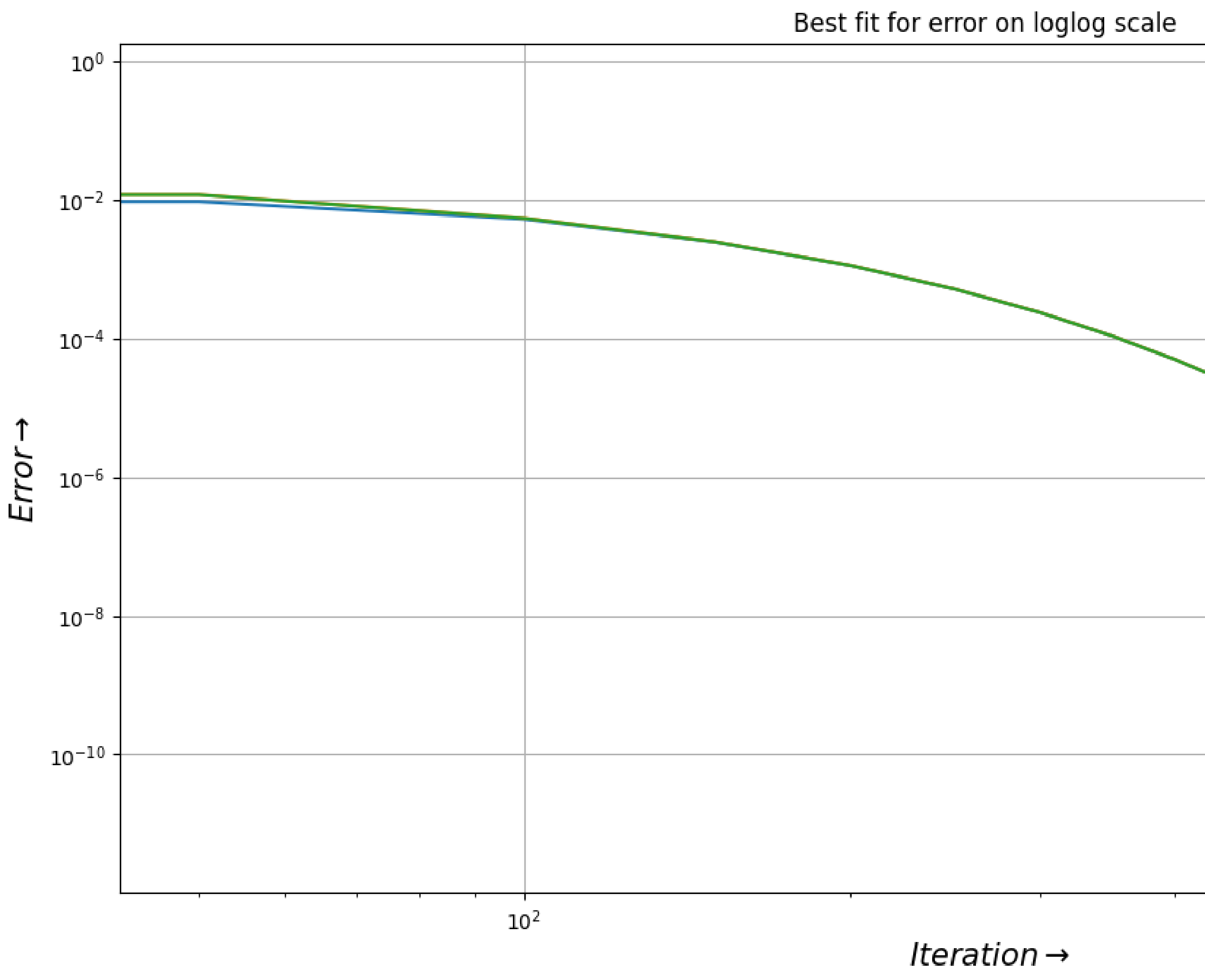

Figure 4: Potting exponential curves

2.6 Plotting Potential

```
# Contour plot of potential
figure(5)
contourf(X,Y,phi)
```

```
plot(ii[0]/Nx-0.48,ii[1]/Ny-0.48,'ro',label="V = 1")


# Surface plot of potential
fig1=figure(6)
ax=p3.Axes3D(fig1, auto_add_to_figure = False)
fig1.add_axes(ax)
```

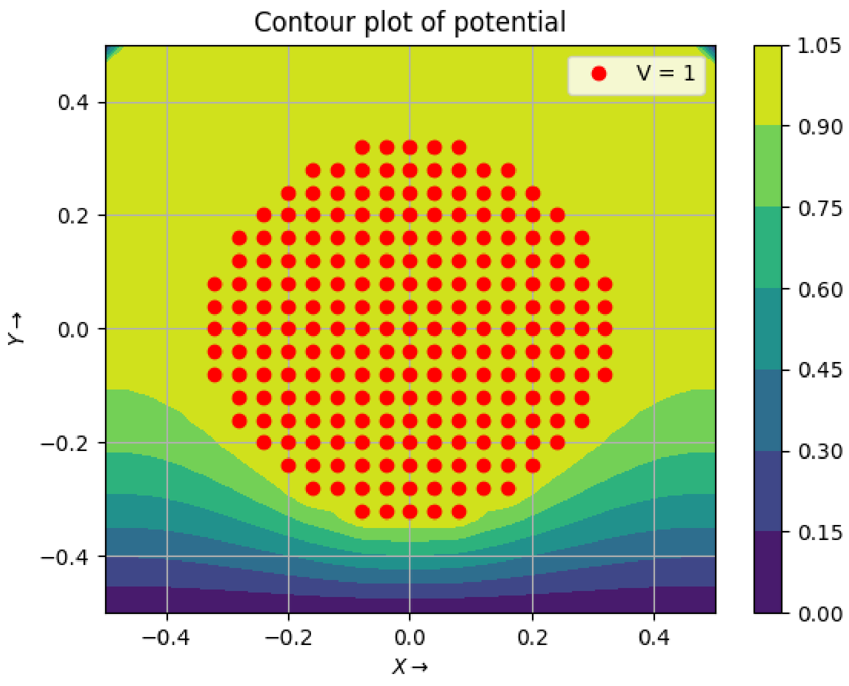

Figure 5: Contour plot of potential

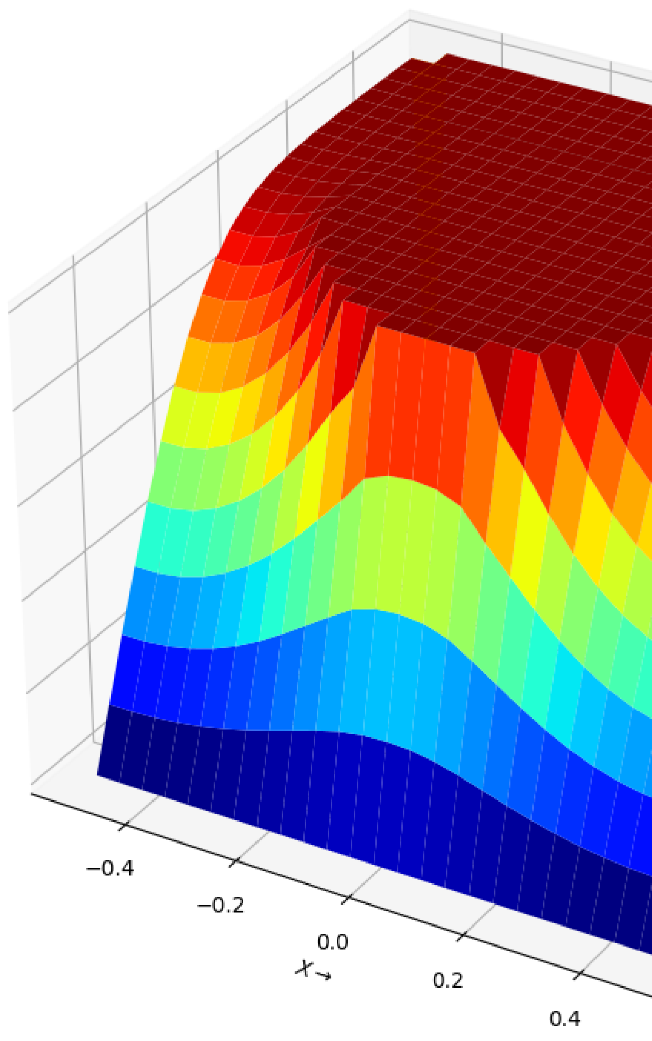

Figure 6: 3D Potential plot

2.7 Vector plot of currents

We use the below equations to calculate current,

$$J_{x,ij} = \frac{1}{2}(\phi_{i,j-1} - \phi_{i,j+1}) \tag{6}$$

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

```
J_x = zeros((Ny, Nx))
J_y = zeros((Ny, Nx))
J_x[1:-1, 1:-1] = 0.5*(phi[1:-1, 0:-2] - phi[1:-1, 2:])
J_y[1:-1, 1:-1] = 0.5*(phi[2:, 1:-1] - phi[0:-2, 1:-1])

#plotting
quiver(X,Y,J_x,J_y)
plot(ii[0]/Nx-0.48,ii[1]/Ny-0.48,'ro')
```

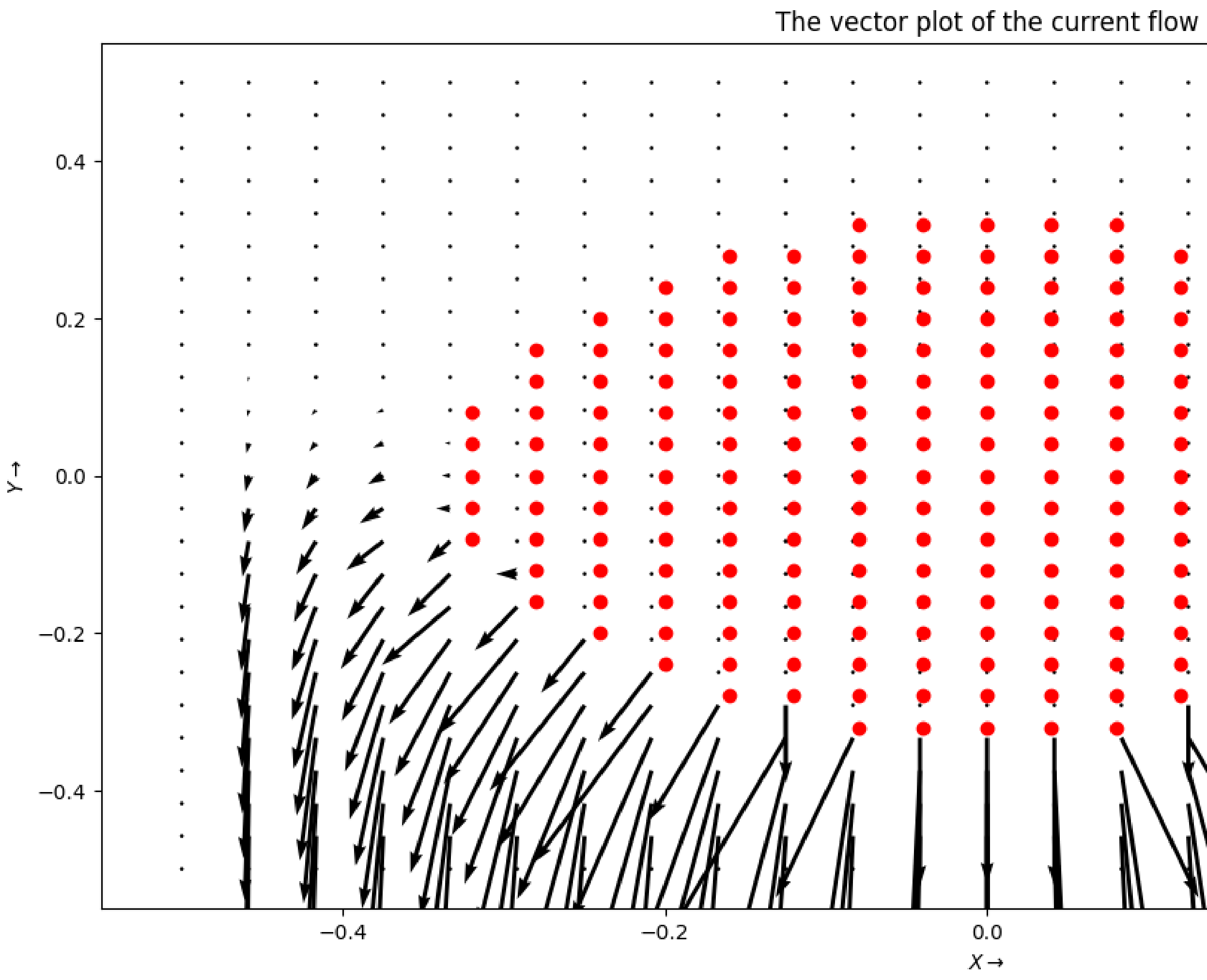

Figure 7: Current vectors

3 Conclusion

- Most of the current is restricted to bottom part of the wire and it is normal to the surface of both wire and metal
- We can vectorize multiple "for" loops to a single line in python
- Also we observed that the decrease in error is very slow after 500 iterations which makes this method inefficient