EE2703 ASSIGNMENT 5

Nandha Varman
EE19B043

1 Outline:

In this assignment, we solve for the currents in a copper plate. We also want to know which part of the resistor is likely to get hottest.

A wire is soldered to the middle of the 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.

2 Theory:

There is a current flow due to the difference in potential. The current at each point can be described by a "current density" $\vec{j}$. This current density is related to the local electric field by the conductivity:

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

Now the electric field is the gradient of potential,

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

By continuity of charge,

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

Combining these equations we obtain,

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

Assuming that our resistor contains a material of constant conductivity, the equation becomes

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

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

$$\nabla^2 \phi = 0$$

The above equation can be written in 2D Cartesian coordinates as a difference equation as follows

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

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

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

Combining this with the corresponding equation for the y derivatives, we obtain

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

Thus, the potential at any point should be the average of its neighbours. This is a very general result and the above calculation is just a special case of it.

So the solution process is obvious. We can guess anything we like for the solution and at each point, replace the potential by the average of its neighbours. We must Keep iterating 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 must just put the value of the potential itself.

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, what this means is the gradient of ϕ should be tangential. This is implemented by requiring that ϕ should not vary in the normal direction.

3 Code:

3.1 Importing necessary modules

```
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
from sys import argv
```

3.2 Parameter definition

```
Nx= 30      # size along x
Ny= 30      # size along y
radius= 8   # radius of central lead
Niter= 4000 # number of iterations to perform

if(len(argv) == 5):    # extracting from optional command line arguments
    Nx = int(argv[1])
    Ny = int(argv[2])
    radius = int(argv[3])
    Niter = int(argv[4])
```

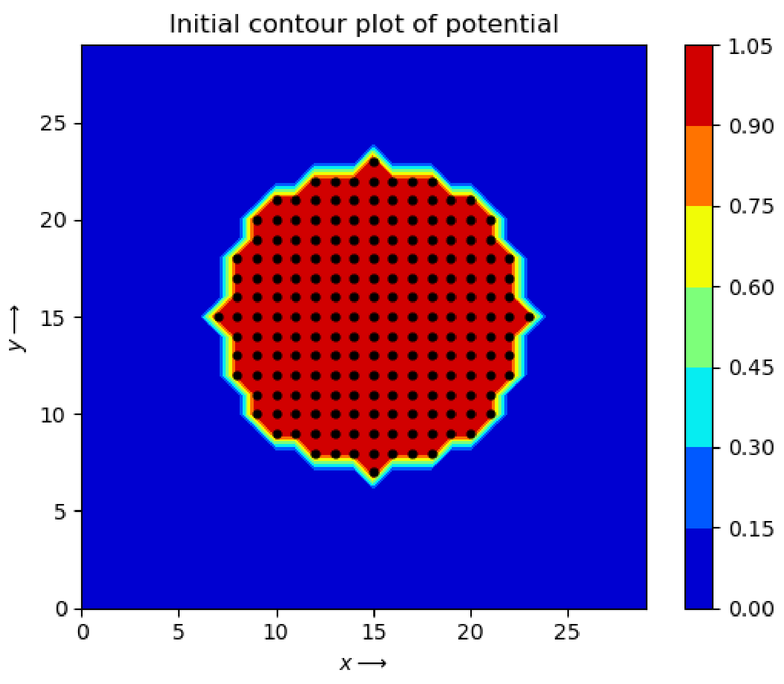
3.3 Allocating the potential array and initializing it

```
phi = zeros((Ny, Nx))   # initializing voltage potential array

y = arange(Ny)
x = arange(Nx)
X,Y=meshgrid(x,y)       # creating mesh grid

ii = where((X-Nx/2)*(X- Nx/2)+ (Y-Ny/2)*(Y-Ny/2)<= radius*radius)
phi[ii] = 1.0           # setting voltge of 1V region

figure(1)
contourf(X, Y, phi, cmap=cm.jet)
axes().set_aspect('equal')
colorbar(orientation = 'vertical')
scatter(x[ii[0]], y[ii[1]], color = 'black', s =12)
title('Initial contour plot of potential')
xlabel('$x\longrightarrow$')
ylabel('$y\longrightarrow$')
show()
```



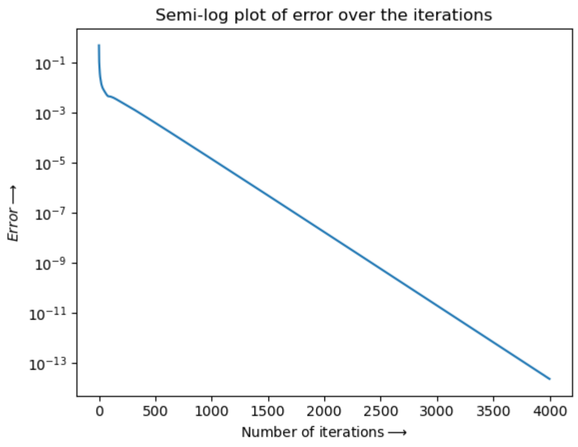
3.4 Iteration for potential and error calculation:

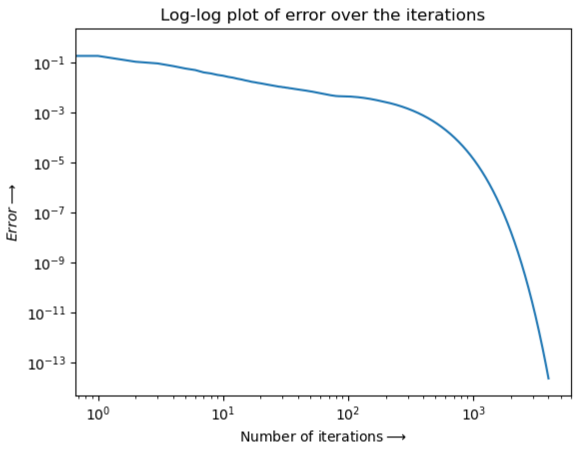
```
errors = zeros(Niter)           # initializing error array
for k in range(Niter):          # updating potential
    oldphi = phi.copy()
    # updating phi array
    phi[1:-1,1:-1]=0.25*(phi[1:-1,0:-2] + phi[1:-1,2:] + phi[0:-2,1:-1] +
    ↪  phi[2:,1:-1])
    # asserting boundaries
    phi[1:-1,0]=phi[1:-1,1]
    phi[1:-1,-1] = phi[1:-1,-2]
    phi[-1,1:-1] = phi[-2,1:-1]
    # setting potential at electrode
    phi[ii] = 1.0
    # error calculation
    errors[k]=(abs(phi-oldphi)).max()
```

3.5 Plotting error over the iterations:

```
figure(2)
semilogy(range(Niter),errors)    # semilog plot of errors
xlabel('Number of iterations$\longrightarrow$')
ylabel('$Error\longrightarrow$')
title('Semi-log plot of error over the iterations')
show()

figure(3)
loglog(range(Niter), errors)    # loglog plot of errors
xlabel('Number of iterations$\longrightarrow$')
ylabel('$Error\longrightarrow$')
title('Log-log plot of error over the iterations')
show()
```





3.5.1 Observations and results:

- We can observe that the semilog plot of error decreases linearly for for higher number of iterations.
- Therefore for large iterations, error decreases exponentially with number of iterations, i.e it is of the form Ae^{Bx}.

3.6 Fitting error over the iterations:

We try to estimate the error over the iterations as an exponential fit

$$y = Ae^{Bx}$$

Taking *log* on both sides, we have

$$logy = logA + Bx$$

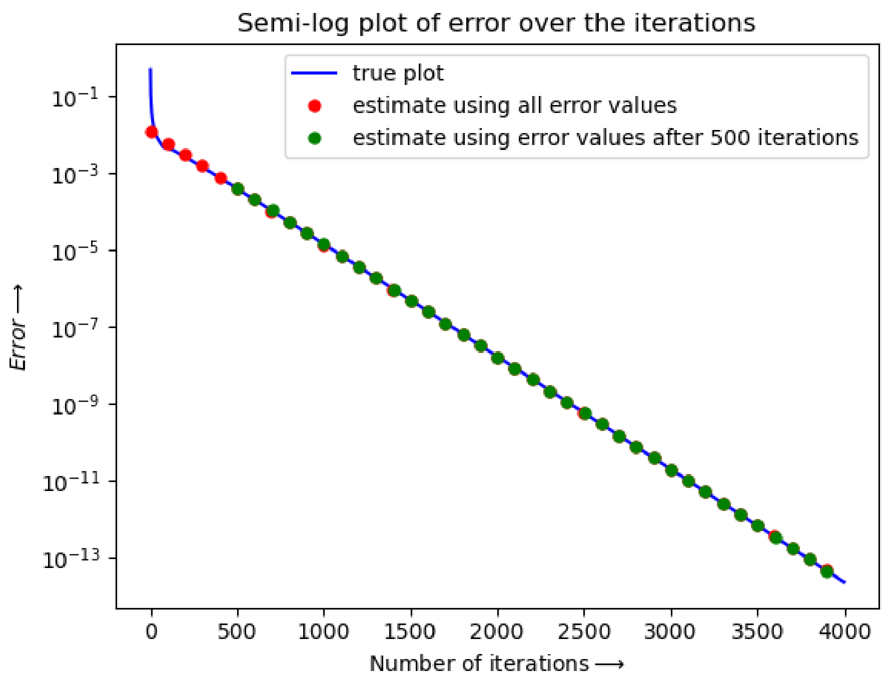
We then use least squares function to find the estimate using all error data, and only the errors after 500 iterations

```
def lstsq_estimate(iterations, y_vec):   # function to fit the data and
↪   return the estimate
    num_iter = len(iterations)
    coeff_matrix = zeros((num_iter,2),dtype = float)
    const_matrix = zeros((num_iter), dtype = float)
    coeff_matrix[:,0] = 1.0
    coeff_matrix[:,1] = iterations
    const_matrix = log(y_vec)
    fit = lstsq(coeff_matrix, const_matrix, rcond = None)[0]
    estimate = coeff_matrix@fit
    return estimate, fit

iterations = array(range(Niter))        # array spanning from 0 to
↪   Niter-1


figure(4)
semilogy(range(Niter),errors, 'b', label = 'true plot')
xlabel('Number of iterations$\longrightarrow$')
ylabel('$Error\longrightarrow$')
title('Log-log plot of error over the iterations')
estimate_all = lstsq_estimate(iterations, errors)[0]
print(lstsq_estimate(iterations, errors)[1])
semilogy(arange(0,Niter,100), exp(estimate_all[::100]), 'or', markersize
↪   = 5, label = 'estimate using all error values')
estimate_after500 = lstsq_estimate(array(iterations[501:]),
↪   array(errors[501:]))[0]
print(lstsq_estimate(array(iterations[501:]), array(errors[501:]))[1])
semilogy(iterations[501::100], exp(estimate_after500[::100]), 'og',
↪   markersize = 5, label = 'estimate using error values after 500
↪   iterations')
```

```
legend(loc = 'upper right')
show()
```



3.6.1 Observations and Results:

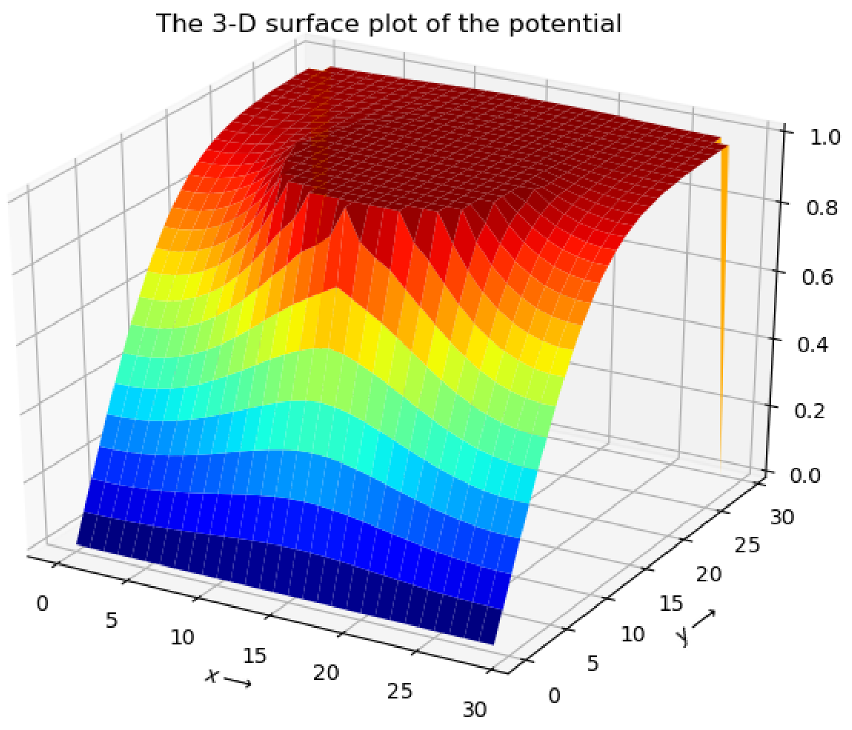
- For small Nx, Ny , the two fits show little difference and almost coincide.
- For larger Nx, Ny (say 100), the estimate using iterations after 500 gives a better fit for the errors.

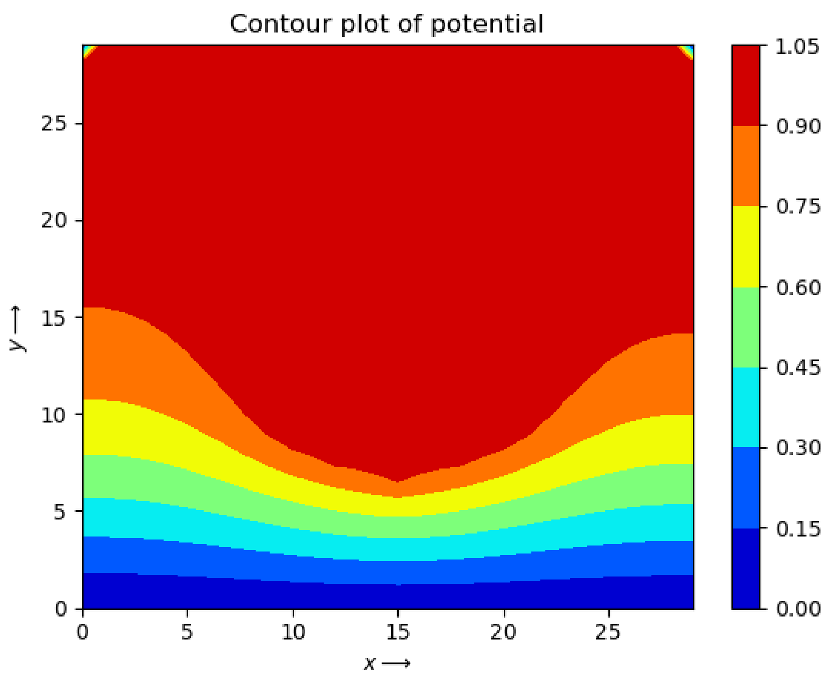
3.7 3D surface plot of potential:

```
figure(5)                          # 3D surface plot of potential
ax=p3.Axes3D(figure(5))
title('3D surface plot and contour plot of potential')
ax.plot_surface(X, Y, phi, rstride=1, cstride=1, cmap=cm.jet)
xlabel('$x\longrightarrow$')
ylabel('$y\longrightarrow$')
```

```
show()

figure(6)                          # contour plot of potential
cs = contourf(X ,Y, phi, cmap=cm.jet)
colorbar(orientation = 'vertical')
title('Contour plot of potential')
xlabel('$x\longrightarrow$')
ylabel('$y\longrightarrow$')
show()
```





3.8 Currents in the copper plate:

To obtain the current densities we must compute the gradient. Since, σ is constant, its actual value does not matter for the shape of the current profile and so we set it to unity.

Our equations are

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

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

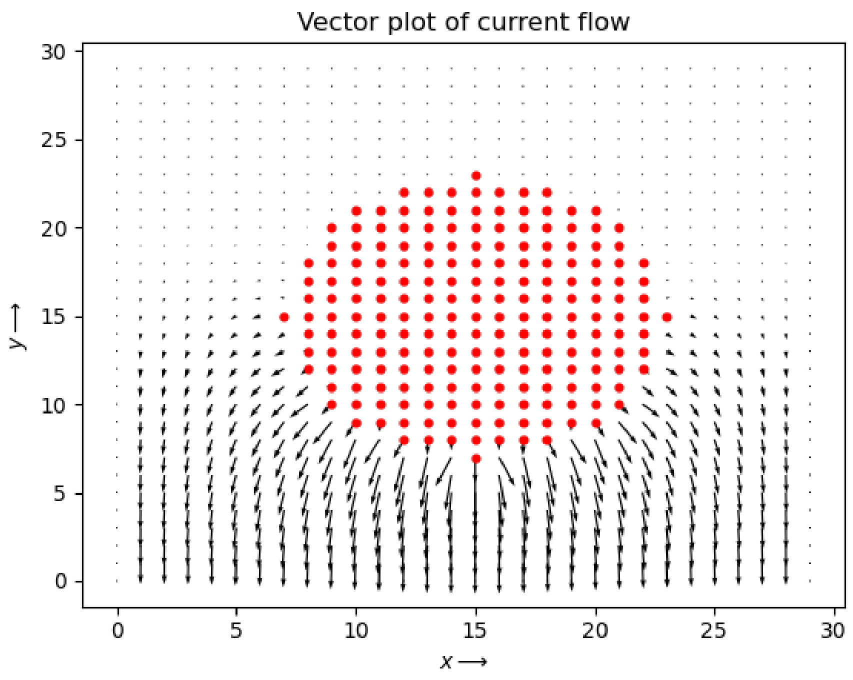
This numerically translates to

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

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

```
figure(7)
Jx = zeros((Nx, Ny))
```

```
Jy = zeros((Nx, Ny))
Jx[1:-1, 1:-1] = (phi[1:-1, 0:-2] - phi[1:-1, 2:])*0.5
Jy[1:-1, 1:-1] = (phi[0:-2, 1:-1] - phi[2:, 1:-1])*0.5
h = quiver(X, Y,Jx[::,::],Jy[::,::], scale = 3)
scatter(x[ii[0]], y[ii[1]], color = 'r', s =12)
title("Vector plot of current flow")
xlabel('$x\longrightarrow$')
ylabel('$y\longrightarrow$')
show()
```



4 Conclusions:

- The current fills the entire cross-section and then flows in the direction of the grounded electrode.
- Most of the current is in the narrow region at the bottom. We also know that the heat generated is due to the Ohmic loss $\vec{J} \cdot \vec{E} = \frac{1}{\sigma}|J|^2$. Hence, this is the part that gets the hottest.

- This method of solving Laplace's Equation is not very good because of the very slow coefficient with which the error reduces.