EE2703 : Applied Programming Lab

G.Rohith Kumar
EE18B008

March 5, 2020

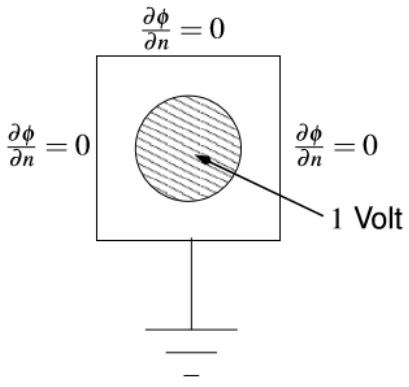
Assignment 5:Resistor Problem

Abstract

We wish to solve for the currents in a resistor. The currents depend on the shape of the resistor.

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 grounded, while the remaining are floating. The plate is 1 cm by 1 cm in size.



By Using Equations of conductivity,Electric field is gradient of potential and continuity

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

$$\vec{E} = -\nabla \phi \tag{2}$$

$$\nabla . \vec{j} = -\frac{\partial \rho}{\partial t} \tag{3}$$

Combining the above three We get,

$$\nabla^2 \phi = \frac{1}{\sigma} \frac{\partial \rho}{\partial t} \tag{4}$$

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

$$\nabla^2 \phi = 0 \tag{5}$$

Solving the laplace equation by transforming it in to differential equation for 2D plane, it can be written out as

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{6}$$

by combining the partial derivatives of phi with respect to x and y as per the above equation we get

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{4} \tag{7}$$

Thus, if the solution holds, 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. Guess anything you like for the solution. At each point, replace the potential by the average of its neighbours. Keep iterating till the solution converges (i.e., the maximum change in elements of is less than some tolerance).

At boundaries where the electrode is present,just put the value of 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 should be tangential. This is implemented by requiring that should not vary in the normal direction.

1. Defining parameters of phi

Code:

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

if(len(sys.argv)==5):
        Nx=int(sys.argv[1])
        Ny=int(sys.argv[2])
        if(int(sys.argv[3])<((int(sys.argv[2])-1)/2)+1):
                radius=int(sys.argv[3])
        else:
                print("wrong input")
                exit()
        Niter=int(sys.argv[4])
else:
        print('taking the default parameters')
        Nx=25;                                  # size along x
        Ny=25;                                  # size along y
        radius=8;                          # radius of central lead
        Niter=1500;                   # number of iterations to perform
```

2. Initializing Potential array phi

Required Conditions: X*X+Y*Y $\leq$ 0.35*0.35

Code:

```
phi=zeros([Ny,Nx])                # phi is potential array
x=linspace(-(Nx-1)/2,(Nx-1)/2,Nx)
y=linspace((Ny-1)/2,-(Ny-1)/2,Ny)
X,Y=meshgrid(x,y)
xy=where(X*X+Y*Y<(radius*radius))
phi[xy]=1         # Making potetial at points where x^2+y^2<c^2
```

3. Updating the Potential for each Iteration

Instead of Using for loop to update potential for each interation, we use vectorization of for loops in python i.e.,
The (i, j)th element of phi[1:Ny-1,1:Nx-1] is phi[i+1,j+1]. The (i, j)th element of phi[1:Ny-1,0:Nx-2] is phi[i+1,j], which is the left neighbour of phi(i+1,j+1). Similarly construct the matrix of right neighbours, top neighbours and bottom neighbours.

Code:

```
def update_phi(phi):
    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])
    return phi
```

4. Asserting Boundary Conditions

Using the same method as in updating potential

Code:

```
def boundaries_phi(phi):                # Boundary Conditions
    phi[:,0]=phi[:,1]
    phi[:,-1]=phi[:,-2]
    phi[0,:]=phi[1,:]
    phi[-1,:]=0
    return phi
```

5. Plotting errors

Error is the maximum of difference between updated potential array and old potential array for each iteration.

Code:

```
errors=zeros(Niter)
for k in range(Niter):
    oldphi=phi.copy()
    update_phi(phi)
    boundaries_phi(phi)
    phi[xy]=1
    errors[k]=((abs(phi-oldphi)).max())
```

Storing all the errors and plotting it will give you a straight line upto 500 iterations and beyond that,you get an exponential variation.
So, let us try to fit the entire errors vector and only the portion beyond 500 in

$$y = Ae^{Bx} \tag{8}$$

By taking log on both sides, we have

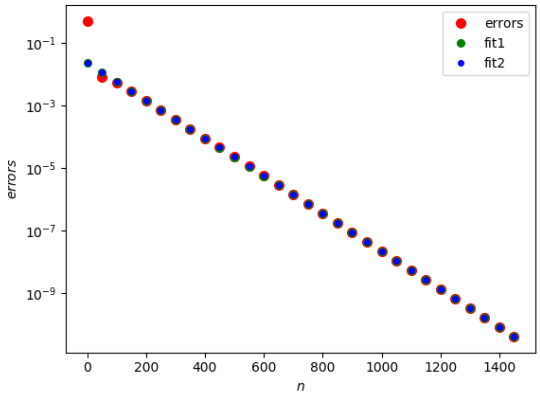
$$logy = logA + Bx \tag{9}$$

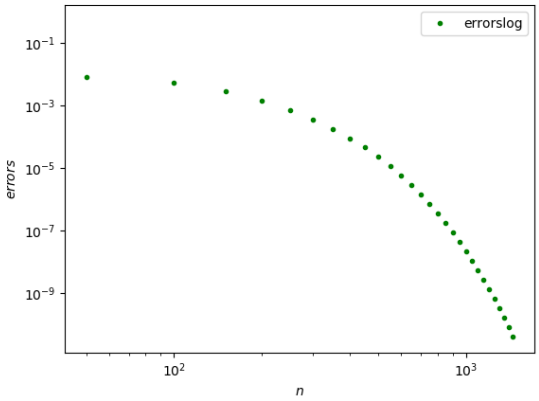
By using lstsq we find the bestfit values of A and B in both cases and substitute them to find the errors in both cases and plot every 50th point of the (errors,errfit1,errfit2).

Code:

```
A=[]
for i in range(Niter):
    A.append([1,i])
p1=lstsq(c_[A],c_[log(errors)],rcond=1)[0]
p2=lstsq(c_[A[500:]],c_[log(errors[500:])],rcond=1)[0]
errfit1=e**dot(c_[A],p1)
errfit2=e**dot(c_[A],p2)
figure(1)
semilogy(c_[A][0:,1:2][::50],errors[::50],'r.',markersize=14,label='errors')
semilogy(c_[A][0:,1:2][::50],errfit1[::50],'g.',markersize=11,label='fit1')
semilogy(c_[A][0:,1:2][::50],errfit2[::50],'b.',markersize=8,label='fit2')
legend(loc='upper right')
```

```
xlabel(r'$n$',size=10)
ylabel(r'$errors$',size=10)
figure(6)
loglog(c_[A][0:,1:2][::50],errors[::50],'g.',label='errorslog')
legend(loc='upper right')
xlabel(r'$n$',size=10)
ylabel(r'$errors$',size=10)
```

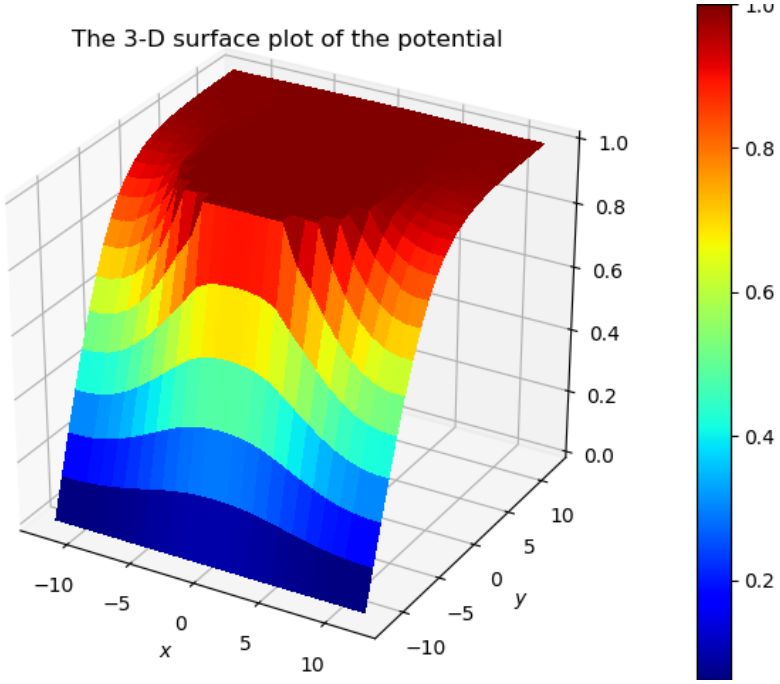




6. Surface Plot of Potential

Code:

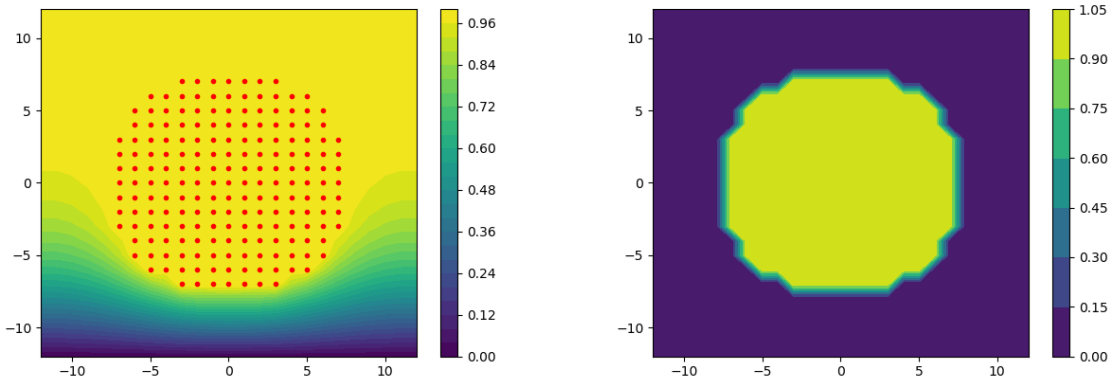
```
ax=p3.Axes3D(figure(2)) # Axes3D is the means to do a surface plot
title('The 3-D surface plot of the potential')
surf = ax.plot_surface(X,Y,phi,rstride=1,cstride=1,cmap=cm.jet,linewidth=0
            ,antialiased=False)                                    # Surface plot
colorbar(surf)
xlabel(r'$x$',size=10)
ylabel(r'$y$',size=10)
```



7. Contour Plot of the Potential

Code:

```
figure(3)
con=contourf(X,Y,phi,Nx)          # Plotting contour of potential phi
plot(X[xy],Y[xy],'r.')
colorbar(con)
figure(4)
con2=contourf(X,Y,electrode)
colorbar(con2)                                    # plotting 1 volt potentia
```



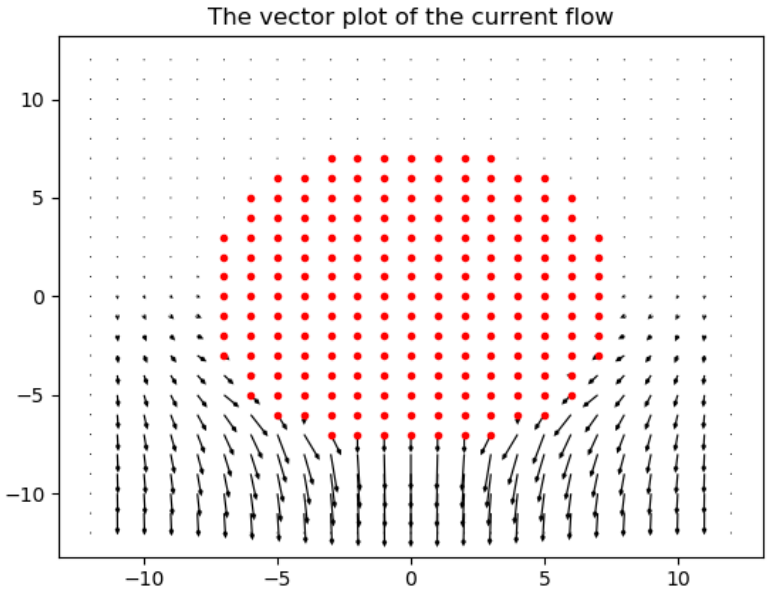
8. Vector Plot of Currents

This requires computing the gradient. The actual value of does not matter to the shape of the current profile, so we set it to unity. Our equations are

$$j_x = \frac{\partial \phi}{\partial x} j_y = \frac{\partial \phi}{\partial y} \tag{10}$$

Create the arrays Jx, Jy. Then call the quiver command:
Code:

```
figure(3)
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[0:-2,1:-1]-phi[2:,1:-1])
figure(5)
title('The vector plot of the current flow')
plot(X[xy],Y[xy],'r.')
quiver(X,Y,-jx,-jy,headlength=3,headwidth=3,scale=4)
show()
```



Conclusion

Inferences from the tasks:-

- Acquired the knowledge of potential and its variation in this particular problem.
- Learnt about the dependency of potential at a point on it's neighbour points.
- Understood the variation of current density with respect to the potential gradient.
- Learnt how to vectorize any equation in python.