Assignment5

EE17B119

March 2019

1 Introduction

The aim of this assignment is to solve for currents in a resistor, by using the method of **finite differences**, which depends on the shape of the resistor and we also want to find out the heat variation

2 Laplace's equation and the method of finite differences

From Laplace's equation, we have

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

For DC currents, the right side is zero.

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

Assuming a 2D situation for our resistor, the laplace equation boils down to

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

The method of finite differences involves considering the region as a tensor, and taking either the forward, backward or central difference about a point, which gives the derivative.

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

On using the method of finite differences for the second derivative, we get:

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

```
#!/usr/bin/python3.5
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
Nx=25 # Size along x
Ny=25 # Size along y
radius= 8# Radius of central lead
Niter=1500 # Number of iterations to perform
errors = zeros(Niter)

def plot_any(y, x=None, fig=1, _ylabel='y', _xlabel='x',
     _title='title', plot_type=1, color='r', l_dots='-',_legend =
     'Curve', save='fig.png'):
    '''
    This function plots any of the function/coefficient plots.
    It takes parameters for the x and y values, labels, titles,
        structure and color.
    '''
    figure(fig) # Choosing figure number
    title(_title) # Plot title
    ylabel(_ylabel) # Setting labels
    xlabel(_xlabel)
    ## To check what kind of plot we need, a regular, semilog or
        loglog
    if plot_type == 1:
        plot(x,y,color+l_dots, label = _legend)
    elif plot_type == 2:
        semilogy(x,y,color+l_dots, label = _legend)
    else:
        loglog(x,y,color+l_dots, label = _legend)
    legend()
    savefig(save)

#Code as given in pdf, to index the resistor
phi = zeros((Ny,Nx), dtype=float)
y = linspace(-0.5,0.5,Ny,dtype = float)
x = linspace(-0.5,0.5,Nx,dtype = float)
Y,X = meshgrid(y,x)

#Code to find the resistor region of the simulation, through which
    the current flows.
ii = where(X*X+Y*Y <= 0.35*0.35)
x_c, y_c = where(X*X+Y*Y <= 0.35*0.35)
phi[ii] = 1
phi_orig = phi.copy()

fig = figure(1)
title('Graph of the resistor potentials initially') # Plot title
ylabel('Length of resistor') # Setting labels
xlabel('Width of resistor')
```

```
imshow(phi)
savefig('phi.png')
```

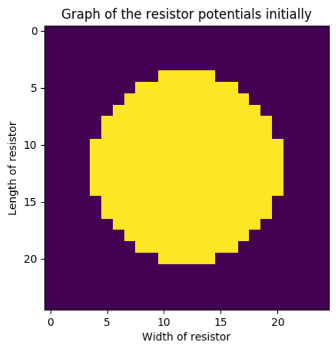

Figure 1: Initial plot of the matrix phi

3 Solving for ϕ after N iterations

We need to keep updating our matrix for ϕ iteration by iteration. We will observe that the iterations start converging to a solution, which is evident by the fact that the error exponentially decreases.
Steps for each iteration:
1) Save each copy of phi
2) Update phi matrix
3) Assert boundary conditions
4) Calculate error

A vectorized implementation to update phi has been implemented, which offers great speed advantage, due to numpy's backend interfacing with C to give high speed ups.

The boundary conditions are:
- The bottom part of the plate, that is grounded is always at 0V
- A Neumann boundary condition is applied at the other edges, where by the derivative is assumed only in the tangential direction, with that in the normal direction being 0.

```
for k in range(Niter):
    #1) Saving a copy of phi
    oldphi = phi.copy()
    #2) Updating 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])
```

```
#3) Asserting the boundary condition
phi[:,0] = phi[:,1]
phi[:,Nx-1] = phi[:,Nx-2]
phi[0,1:-1] = phi[1,1:-1]
phi[Ny-1,:] = 0
# Asserting the source condition
phi[ii] = 1
#4) Finding the error
errors[k] = abs(phi - oldphi).max()
```

The central core of the conductor is maintained at 1V

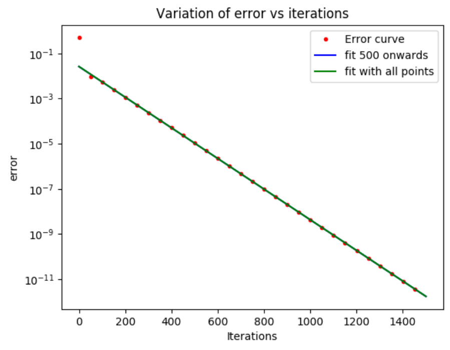

Figure 2: Plot of error variation with number of iterations

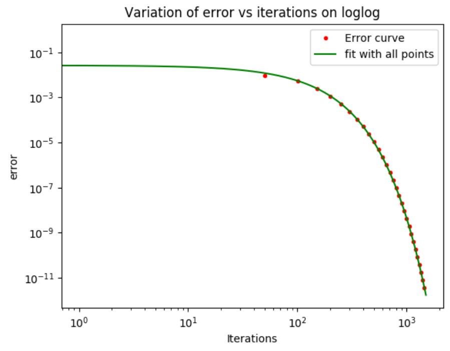

Figure 3: Plot of error variation with number of iterations on Loglog plot

4 Stopping condition

The error can be modelled as Ae^{Bk}, where k is the iteration number. The error is found as the maximum absolute between the changing values of the matrix

phi.
$error_k = max(abs(phi - oldphi))$

The values of error are fit to the above function, Ae^{Bk} A and B are determined through a least square fit. Found values of A and B are (0.026559901280721426, -0.01564226175468142)

This fit is done twice, once for all the values of error obtained, one just for the values from iteration 500 on wards, as the plot is exponential only or higher values of the iteration number.
Both The methods yield almost the same value for A and B.

```
    def fit_error(error, x2):
        # lstsq, Ax=b, x = logA,B, A is [1 x], b is logy
        matA =
            concatenate((ones((len(error),1)),x2.reshape((len(x2),1))),
            axis = 1)
        matb = log(error)
        logA_B = lstsq(matA,matb)[0]
        #logA_B contains the required coefficients for the error fit.
        return exp(logA_B[0]),logA_B[1]
```

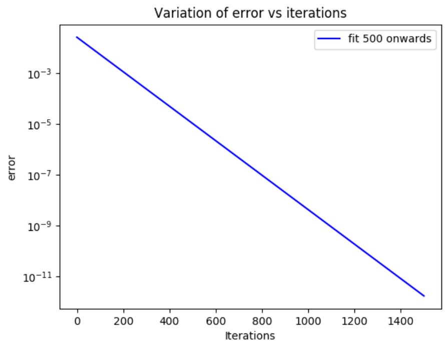

Figure 4: Plot of error variation with number of iterations

5 Surface and contour Plots

A 3D surface plot and a contour plot of the variation of phi have been plotted.

```
#3D plot
fig1=figure(3)   # open a new figure
ax=p3.Axes3D(fig1) # Axes3D is the means to do a surface plot
title('The 3-D surface plot of the potential')
ylabel('Length of resistor') # Setting labels
```

```
xlabel('Width of resistor')
surf = ax.plot_surface(Y, X, phi.T, rstride=1, cstride=1, cmap=cm.jet)
savefig('3d.png')

fig = figure(4)
#Contour plot of potential.
title('Contour plot of potential') # Plot title
ylabel('Length of resistor') # Setting labels
xlabel('Width of resistor')
cont = contourf(x[::-1],y[::-1],phi,10)
colorbar()
plot((x_c-Nx/2)/(Nx*1.0),(y_c-Ny/2)/(Ny*1.0),'ro')
savefig('pot_cont.png')
```

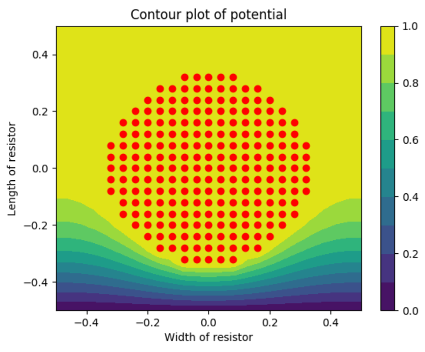

Figure 5: Contour plot of the potential

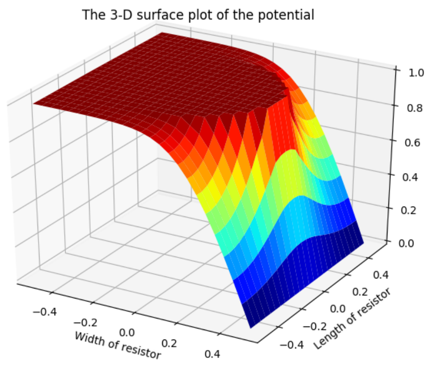

Figure 6: 3D plot of the potential after simulation

6 Vector plot of Current Density

We know

$$J_x = -\partial\phi/\partial x \; J_y = -\partial\phi/\partial y$$

So, using the method of finite differences, this transforms to

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

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

Using these values obtained, a quiver plot is made to visualize the currents.

```
fig = figure(5)
## Finding the current derivatives
Jx = 0.5*(phi[0:-2,1:-1]-phi[2:,1:-1])
Jy = 0.5*(phi[1:-1,2:]-phi[1:-1,0:-2])
title('Quiver plot of current') # Plot title
ylabel('Length of resistor') # Setting labels
xlabel('Width of resistor')
quiver(Y[1:-1,1:-1],-X[1:-1,1:-1],Jy[:,::-1],-Jx)
plot((x_c-Nx/2)/(Nx*1.0),(y_c-Ny/2)/(Ny*1.0),'ro')
savefig('quiver.png')
```

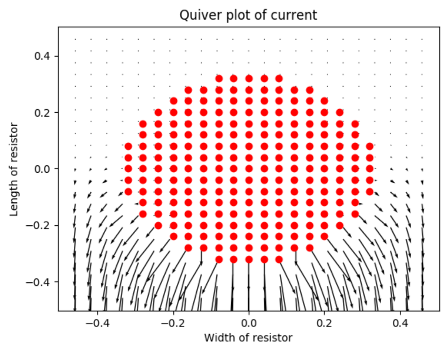

Figure 7: Quiver plot of the currents

7 Extra work: Plot of temperature variation

A similar equation is solved, only this time the RHS is not 0, it is the sum of squares of all the elements of the current matrives just found.
The method is exactly the same

1) Save a copy of the temperature matrix
2) Update the temperature matrix
3) Assert Boundary conditions

Here, the boundary conditions are 300K at the core of the conductor and at ground.

```
##Solving for the temperature
temp = 300*ones((Ny,Nx))
for i in range(Niter):
    #1) Save a copy of temp
    oldtemp = temp.copy()
    #2) Update temp
    temp[1:-1,1:-1] = 0.25*(temp[1:-1,0:-2] + temp[1:-1,2:] +
         temp[0:-2,1:-1] + temp[2:,1:-1] + Jx**2 + Jy**2)
    #3) Assert boundary conditions
    temp[:,0] = temp[:,1]
    temp[:,Nx-1] = temp[:,Nx-2]
    temp[0,1:-1] = temp[1,1:-1]
    temp[Ny-1,:] = 300
    #4) Assert source condition
    temp[ii] = 300
fig = figure(6)
#Contour plot of the temperature variation
cont = contourf(x[::-1],y[::-1],temp,10)
title('Contour plot of Temperature') # Plot title
ylabel('Length of resistor') # Setting labels
xlabel('Width of resistor')
plot((x_c-Nx/2)/(Nx*1.0),(y_c-Ny/2)/(Ny*1.0),'ro')
colorbar()
savefig('temp.png')
show()
```

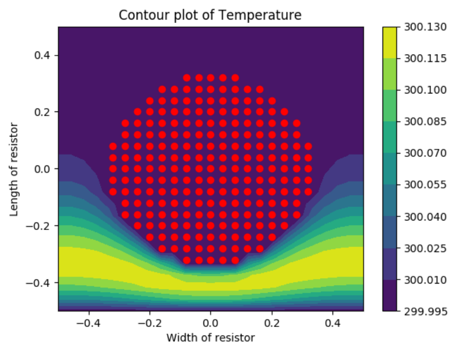

Figure 8: Plot of error variation with number of iterations

8 Conclusion

We have used the method of finite differences to solve the Laplace equation. Owing to the low rate of decrease of the error, this method is deprecated, and known as one of the worse methods. We have studied contour and 3D plots for the same. Analysing the quiver plot confirms that current is normal to the surface of the conductor, both the wire and metal plate.