EE2703: Applied Programming Lab

Assignment # 5

Sourav Sahoo, EE17B040

March 2, 2019

Contents

1 Introduction

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.[1] In this assignment we intend to find the current density and potential gradient in a resistor.

1.1 Some Important Formulae

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

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

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

$$\nabla \cdot (-\sigma \nabla \phi) = -\frac{\partial \rho}{\partial t} \tag{4}$$

$$\nabla^2 \phi = \frac{1}{\sigma}\frac{\partial \rho}{\partial t} = 0 \; for \; DC \; currents \tag{5}$$

2 Initialization

Import the following libraries.

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

The size of the plate and the radius and the number of iterations are taken from the input in command line.Then it is normalized to between (-1,1) in X-axis and Y-axis.

```
if len(sys.argv) != 5:
    print("Incorrect format.The correct format is $ python3 assign5.py Nx Ny Radius Niter")
    print("For best results, 40 <= Nx <= 200,\
            40 <= Nx <= 200,Nx = Ny,Niter = 50*|Nx|,Radius = 0.35*Nx")
    sys.exit()
else:
    Nx,Ny,Radius,Niter = list(map(int,sys.argv[1:]))
    if 2*Radius > Nx:
        print("Too large radius !!")
        sys.exit()

V = np.zeros((Nx,Ny))
x = np.linspace(-1,1,Nx)
y = np.linspace(-1,1,Ny)
Y,X = np.meshgrid(y,x)
Rad_norm = (Radius*2)/Nx
```

The circular electrode is present in the central position. The coordinates are collected and the initial potential is plotted.

```
ii = np.where(X*X + Y*Y < Rad_norm*Rad_norm)
V[ii] = 1.0
fig = plt.figure(figsize=(8,8))
plt.grid()
```

[1] https://en.wikipedia.org/wiki/Resistor

```
plt.scatter(ii[0]-Nx/2,ii[1]-Ny/2,s=4,color='r',marker='o')
plt.title('Initial Potential')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim((-Nx/2,Nx/2))
plt.ylim((-Ny/2,Ny/2))
plt.legend(['V = 1'])
plt.show()
```

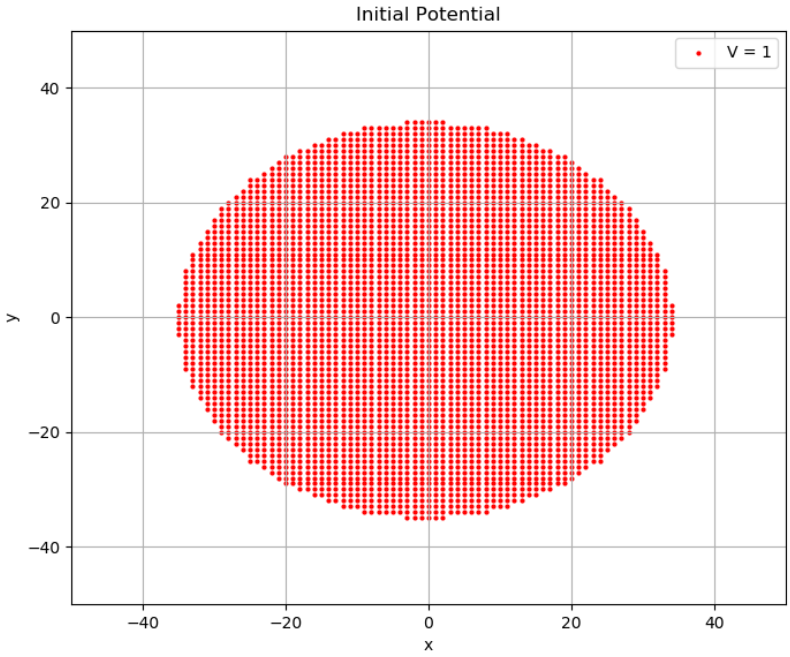

Figure 1: Initial Potential on the plate

3 Performing the Iterations

As the number of iterations and the dimension of the matrix is high,use of vectorized code is greatly required.The error in each iteration is stored in an array.

```
error = np.zeros((Niter,))
for k in range(Niter):
V_old = V.copy()
V[1:-1,1:-1] = 0.25*(V[1:-1,0:-2] + V[1:-1,2:] + V[0:-2,1:-1] + V[2:,1:-1])
V[0,1:-1],V[-1,1:-1],V[:,-1] = V[1,1:-1],V[-2,1:-1],V[:,-2]
V[ii] = 1.0
error[k] = np.max(abs(V - V_old))
```

4 Error Analysis

4.1 Error Plots

The errors are first plotted in log-log scale ,then in semilog scale for all iterations then after 20% of the iterations is completed. Then using `np.linalg.lstsq` the curves are fitted to a linear plot.

```
log_error = np.log(error)
error_new = log_error
n_iter = np.arange(1,Niter+1)

plt.figure(figsize=(8,8))
plt.grid()
plt.title('Error in loglog scale for all iterations')
plt.ylabel('Error in loglog scale')
plt.xlabel('number of iterations')
plt.plot(np.log(n_iter),error_new,'r')
plt.legend(['actual'])
plt.show()

ones = np.ones_like(n_iter)
n_iter = np.vstack((ones,n_iter)).T

plt.figure(figsize=(8,8))
plt.grid()
plt.title('Error in semilog scale for all iterations')
plt.ylabel('Error in semilog scale')
plt.xlabel('number of iterations')
plt.plot(n_iter,error_new,'r')
A,_,_,_ = np.linalg.lstsq(n_iter,error_new,rcond=-1)
plt.plot(n_iter,n_iter.dot(A),color='b')
plt.xlim((0,Niter))
plt.legend(['actual','reconstructed'])
plt.show()

error_new = log_error[int(Niter*0.2):]
n_iter = np.arange(int(Niter*0.2),Niter)
ones = np.ones_like(n_iter)
n_iter = np.vstack((ones,n_iter)).T

plt.figure(figsize=(8,8))
plt.grid()
plt.title('Error in semilog scale after {} iterations'.format(int(Niter*0.2)))
plt.ylabel('Error in semilog scale')
plt.xlabel('number of iterations')
plt.plot(n_iter,error_new,'r')
A,_,_,_ = np.linalg.lstsq(n_iter,error_new,rcond=-1)
plt.plot(n_iter,n_iter.dot(A),color='b')
plt.xlim((Niter*0.15,Niter*1.05))
plt.legend(['actual','reconstructed'])
plt.show()
```

4.2 Error Calculation

The errors are approaches he form $e_k = Ae^{bk}$ as k increases. Taking log on both sides

$$log(e_k) = log(A) + bk \tag{6}$$

When all the iterations are considered,the values of A and b as follows:

$$A = 0.00158520773$$

$$b = -9.02665 \times 10^{-4}$$

When only the iterations after the 1000 th one are considered (for Nx = 100),the values of A and b as follows:

$$A = 0.00134695397$$

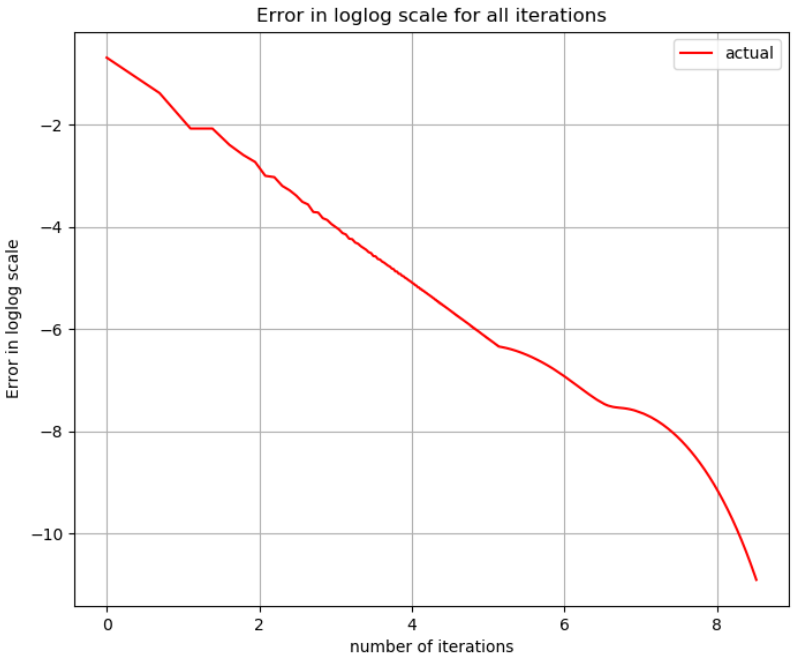

Figure 2: Errors in log-log scale

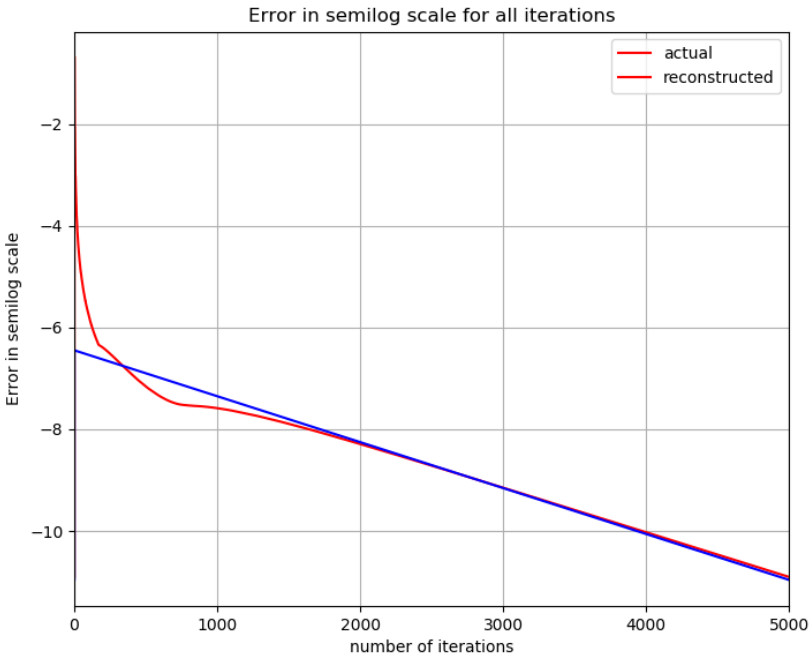

Figure 3: Error in semilog scale for all iterations along with the linear plot from lstsq method

$$b = -8.52777 \times 10^{-4}$$

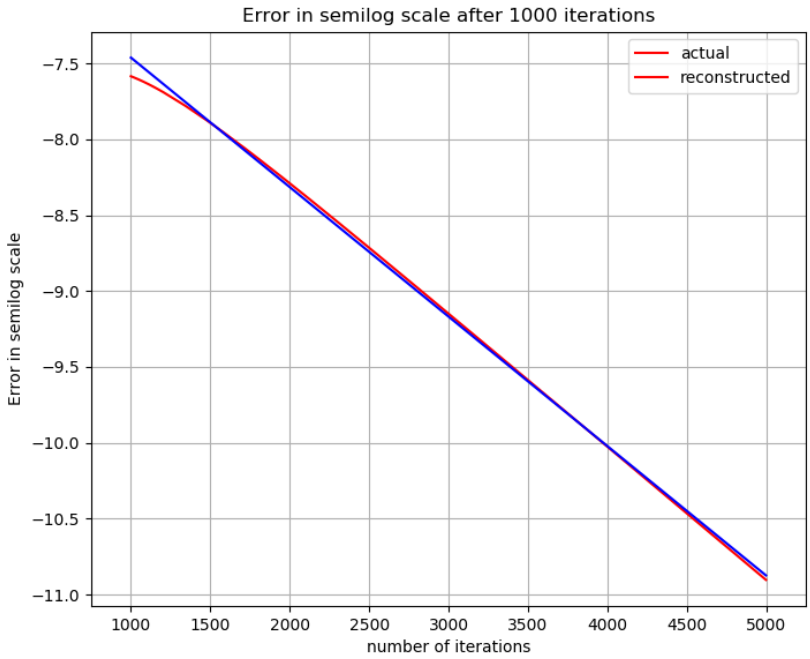

Figure 4: Error in semilog scale after 1000 iterations along with the linear plot from lstsq method

5 Further Analysis

5.1 Surface Plot of Potential

The surface plot of the potential is plotted by executing the following code:

```
fig = plt.figure(figsize=(10,8))
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, V,cmap = cm.jet,linewidth=0, antialiased=False)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.title('3-D Surface plot of the potential')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

5.2 Contour Plot of Potential

The contour plot of the potential is plotted by executing the following code:

```
plt.figure(figsize=(8,8))
plt.grid()
plt.title('Contour plot of the potential')
cp = plt.contour(X,Y,V,20)
levels = np.linspace(0,1,11)
plt.clabel(cp,levels,inline=True,fontsize = 10)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

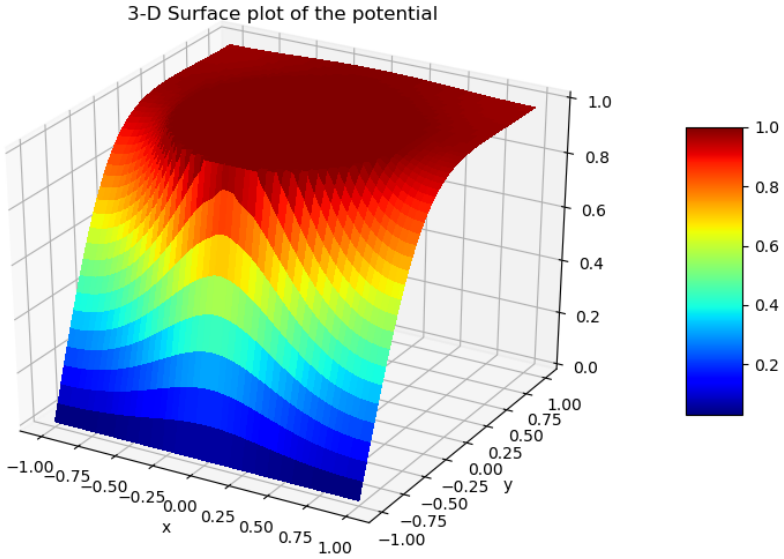

Figure 5: Surface plot of the potential

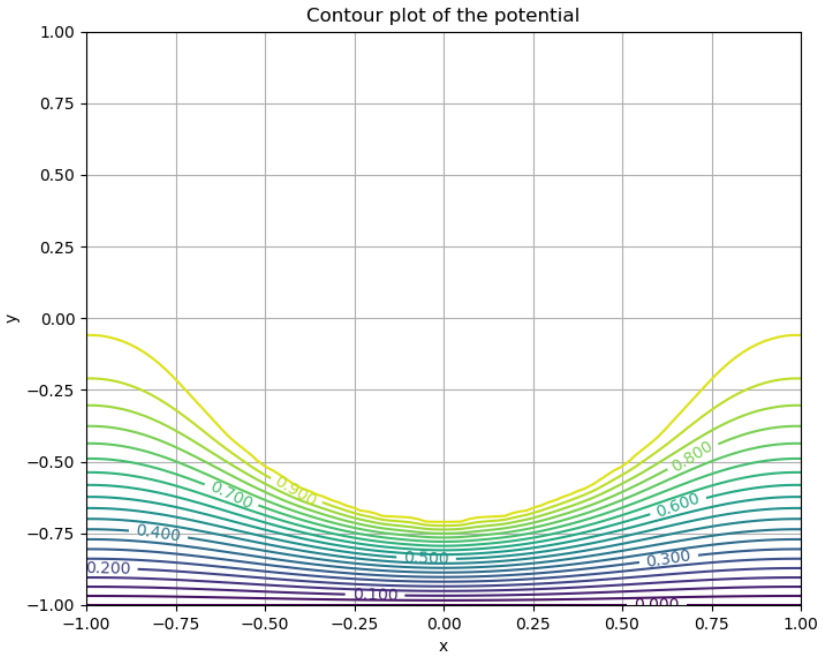

Figure 6: Contour plot of the potential

5.3 Vector Plot of Current

If we consider $\sigma = 1$,

$$\vec{j_x} = -\frac{\partial \phi}{\partial x} \tag{7}$$

$$\vec{j_y} = -\frac{\partial \phi}{\partial y} \tag{8}$$

Using the above equations the current density can be calculated executing the following code. Then `quiver` command is used to do a vector plot of current flowing in the plate.

```
Jx,Jy = np.zeros_like(V),np.zeros_like(V)
plt.figure(figsize=(8,8))
plt.grid()
Jy[:,1:-1] = 0.5*(V[:,:-2] - V[:,2:])
Jx[1:-1] = 0.5*(V[:-2] - V[2:])
plt.scatter((ii[0]-Nx/2)*(2/Nx),(ii[1]-Ny/2)*(2/Ny),s=4,color='r',marker='o')
plt.quiver(X,Y,Jx,Jy)
plt.title('Quiver plot of the current densities')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(['Electrode','Current density'])
plt.show()
```

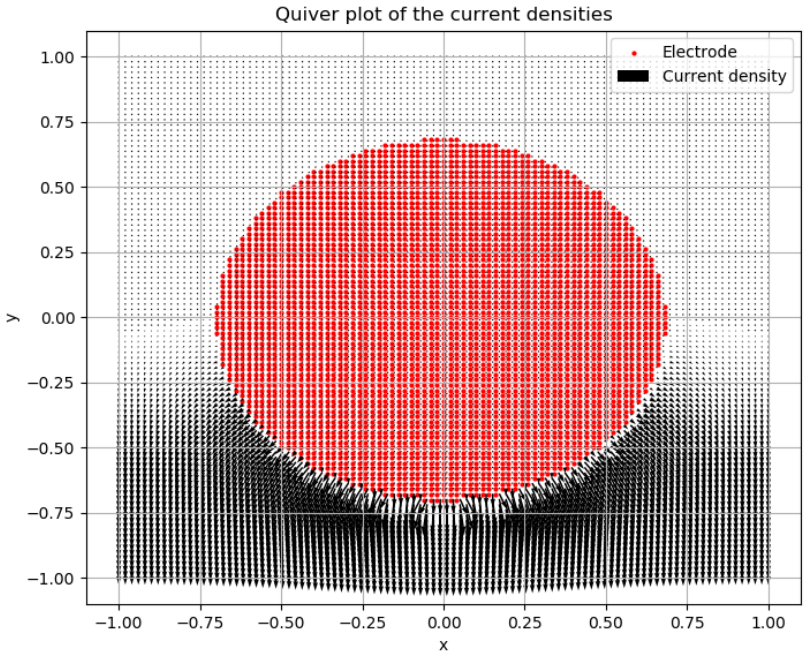

Figure 7: Quiver plot of current

6 Additional

The heat generated is given by:

$$q = \vec{j} \cdot \vec{E} = \frac{1}{\sigma}\vec{j} \cdot \vec{j} = \frac{1}{\sigma}|j|^2 = |j|^2 \; if \; \sigma = 1 \tag{9}$$

The code for the same is :

```
J_sq = Jx**2 + Jy**2
plt.figure(figsize=(8,8))
plt.grid()
```

```
plt.title('Contour plot of the heat generated')
cp = plt.contour(X,Y,J_sq,10)
plt.clabel(cp,inline=True,fontsize = 10,colors='r')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

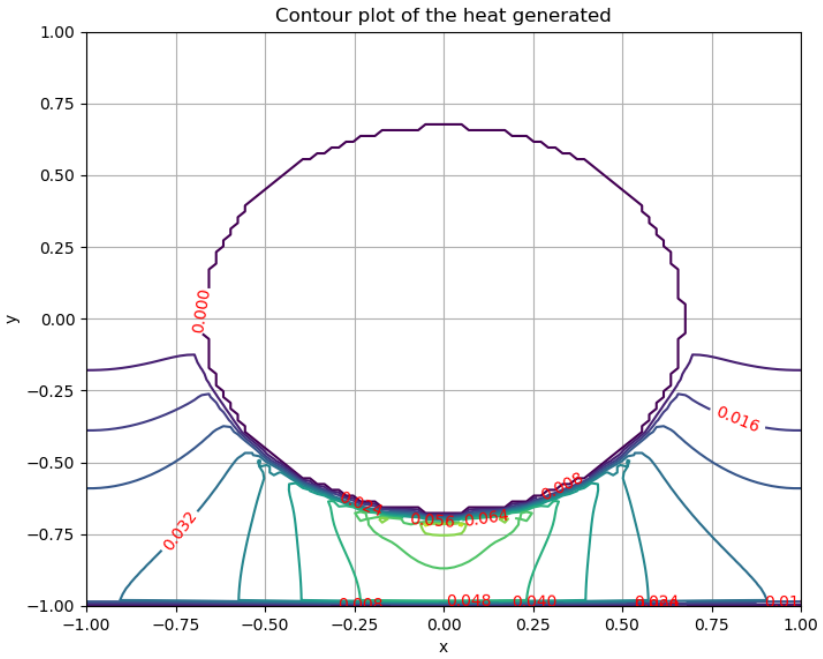

Figure 8: Contour plot of the heat generated

We see that the maximum heat is generated in the region where current density is maximum i.e just under the electrode towards the the ground.

7 Conclusion

In this assignment, we analyzed current densities, potential gradient in a resistor and how they depends on its geometry. We also found out the regions that get heated up the maximum when current flows in a resistor.