LAB 4

HARSHA VARDHAN EE17B061

March 4, 2019

1 visualising Equi-potential lines

we find the potential distribution using repeated implementation of the equation//

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

repeating the above equation for a very large number of times will reduce the error in potential to zero.But the number of iterations required for a reasonably low error increase with number of points(dimension of the matrix). for this reason this a very inefficient method to compute Laplacian.

```
for i in range(Niter):
    oldphi = phi.copy()
    phi[1:-1,1:-1] = (phi[1:-1,:-2]+phi[1:-1,2:]+phi[:-2,1:-1]+phi[2:,1:-1])/4
    phi[ii]=1
    phi[:,0],phi[:,-1] = phi[:,1],phi[:,-2]#neumann conditions at sides
    phi[0],phi[-1]=0,phi[-2]#neumann condition a top and ground at bottom
    errors[i] = np.max(np.abs(phi-oldphi))

    Contour1 = plt.contour(phi)#Equipotential lines for Voltage
plt.clabel(Contour1,inline=1)
```

The Equipotential lines are shown in the following figure.The red dots represent the points inside the radius of the soldered wire.**Neumann** boundary conditions(i.e, $\frac{\partial\phi}{dn} = 0$) are enforced at the sides of the plates that are not grounded.

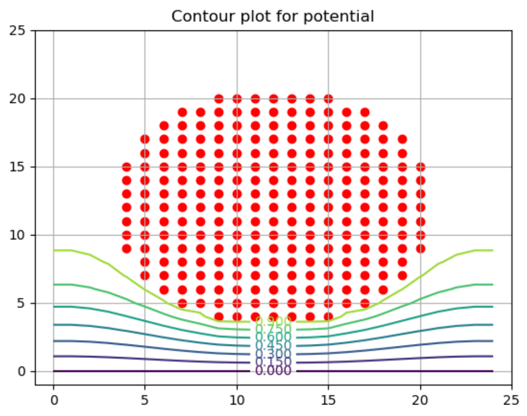

Figure 1: plot3

the current density is calculated assuming $\sigma = 1$.

$$J_x = -\frac{\partial \phi}{\partial x} \tag{2}$$

$$J_y = -\frac{\partial \phi}{\partial y} \tag{3}$$

```
Jx,Jy = np.zeros((Ny,Nx)),np.zeros((Ny,Nx))
Jx[1:-1,1:-1] = -(phi[1:-1,2:]-phi[1:-1,:-2])/2
Jy[1:-1,1:-1] = (phi[:-2,1:-1]-phi[2:,1:-1])/2
plt.quiver(Jx,Jy,width=0.002,scale=1/0.25)
```

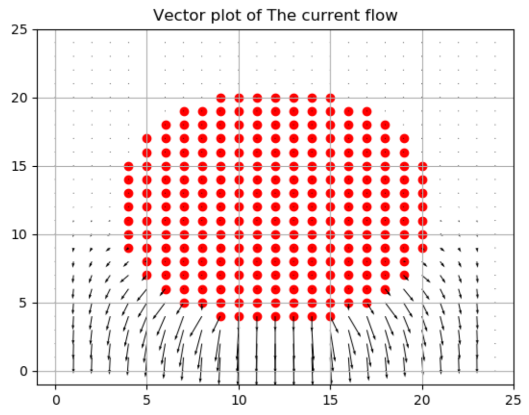

Figure 2: plot3

3d representation of potential on the plate.

```
fig3 = plt.figure(3)
ax = p3.Axes3D(fig3)
plt.title('Surface potential')#plt.zlabel('potential')
surface = ax.plot_surface(X1,Y1,phi,rstride=1,cstride=1,cmap=plt.cm.jet)
```

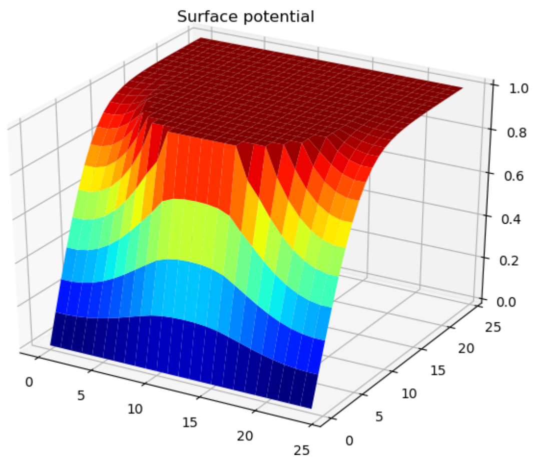

Figure 3: plot3

2 linear fitting for error over iterations

error is varying linearly after around 200 iterations and the fit(coefficients A,B) from overall data and the data after 500 iters is very close.

```
    #error fitting
if Niter>500:
    A1 = lstsq(np.c_[np.ones((Niter-500,1)),np.arange(500,Niter)],np.log(errors[500:]))[0]
    A2 = lstsq(np.c_[np.ones((Niter,1)),np.arange(Niter)],np.log(errors))[0]
    fit1 = np.exp(A1[0]+A1[1]*np.arange(500,Niter))
    fit2 = np.exp(A2[0]+A2[1]*np.arange(Niter))
    # A2 = lstsq(Iters,errors)
    print(A1,A2)

    fig, ax = plt.subplots(1,2)
    plt.title('Error Vs N of iters in loglog and semilogy')

    for data in ('errors','fit2'):
        ax[0].semilogy(eval(data),label=data)
        ax[1].loglog(eval(data),label=data)
    ax[0].semilogy(np.arange(500,Niter),fit1,label='fit_after_500')
    ax[1].loglog(np.arange(500,Niter),fit1,label='fit_after_500')
    for i in range(2):
        ax[i].legend()
```

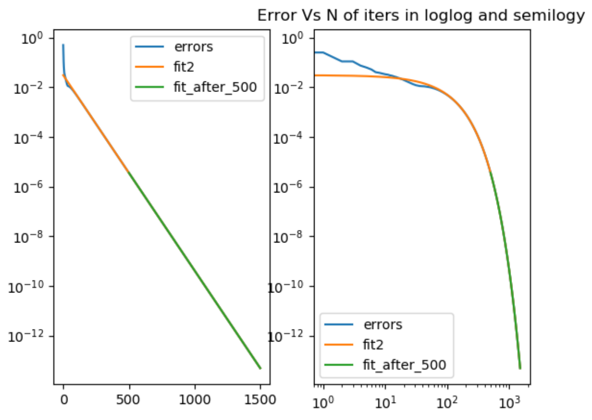

Figure 4: plot4

code

```
import matplotlib.pyplot as plt
import numpy as np
import mpl_toolkits.mplot3d.axes3d as p3
from scipy.linalg import lstsq
from sys import argv
"""
plate is 1cm,radius of wire 0.35mm
shape of the matrix should be (Ny,Nx) because
no of (rows,cols) represent the (height,width) of the plate
"""
argv0 = list(map(int,argv[1:]))

if len(argv0)==4:
    Nx,Ny,radius,Niter=argv0
else:
    Nx,Ny,Niter = (25,25,1500) if len(argv0)==0 else argv0
    radius = min(Ny,Nx)*0.35#scaled radius
```

```
print(radius)
phi = np.zeros((Ny,Nx))

# to set the ends for points under wire_radius calculation
ends = lambda n: (-int(n//2),int(n//2)) if n%2 else (-int(n/2)+0.5,int(n/2)-0.5)

x,y = np.linspace(*ends(Nx),Nx),np.linspace(*ends(Ny),Ny)
Y,X = np.meshgrid(y,x)
X1,Y1 = np.meshgrid(np.arange(Ny),np.arange(Nx))
ii = np.where(X*X+Y*Y<=radius*radius)
phi[ii] = 1
errors = np.zeros(Niter)
"""
I considered plate and the matrix as mirror images about a line || to x-axis at
the middle of the plate. so conditions are applied accordingly to the matrix
V(normal) gradient is zero at sides and bottom and is zero at top(for the matrix
"""
for i in range(Niter):
    oldphi = phi.copy()
    phi[1:-1,1:-1] = (phi[1:-1,:-2]+phi[1:-1,2:]+phi[:-2,1:-1]+phi[2:,1:-1])/4
    phi[ii]=1
    phi[:,0],phi[:,-1] = phi[:,1],phi[:,-2]#neumann conditions at sides
    phi[0],phi[-1]=0,phi[-2]#neumann condition a top and ground at bottom
    errors[i] = np.max(np.abs(phi-oldphi))

#currents
Jx,Jy = np.zeros((Ny,Nx)),np.zeros((Ny,Nx))
Jx[1:-1,1:-1] = -(phi[1:-1,2:]-phi[1:-1,:-2])/2#(-dphi/dx)check gradient in cont
Jy[1:-1,1:-1] = (phi[:-2,1:-1]-phi[2:,1:-1])/2#upper - lower in matrix => lower

#fig1 code
Contour1 = plt.contour(phi)#Equipotential lines for Voltage
plt.clabel(Contour1,inline=1)

for i in (1,2):
    #common plot for figs 1 and 2
    plt.figure(i)
    plt.title('Contour␣plot␣for␣potential' if i==1 else 'Vector␣plot␣of␣The␣curr
    plt.scatter(ii[0],ii[1],c='r')
    plt.xlim([-1,Nx])
    plt.ylim([-1,Ny])
    plt.grid(b=True,which='both',axis='both')

#fig2 code plt from 0 to Nx-1(y-axis) and 0 to Ny-1(x-axis)
plt.quiver(Jx,Jy,width=0.002,scale=1/0.25)
```

```
#3D representation of potential
fig3 = plt.figure(3)
ax = p3.Axes3D(fig3)
plt.title('Surface potential')#plt.zlabel('potential')
surface = ax.plot_surface(X1,Y1,phi,rstride=1,cstride=1,cmap=plt.cm.jet)

#error fitting
if Niter>500:
    A1 = lstsq(np.c_[np.ones((Niter-500,1)),np.arange(500,Niter)],np.log(errors[
    A2 = lstsq(np.c_[np.ones((Niter,1)),np.arange(Niter)],np.log(errors))[0]
    fit1 = np.exp(A1[0]+A1[1]*np.arange(500,Niter))
    fit2 = np.exp(A2[0]+A2[1]*np.arange(Niter))
    # A2 = lstsq(Iters,errors)
    print(A1,A2)

    fig, ax = plt.subplots(1,2)
    plt.title('Error Vs N of iters in loglog and semilogy')

    for data in ('errors','fit2'):
        ax[0].semilogy(eval(data),label=data)
        ax[1].loglog(eval(data),label=data)
    ax[0].semilogy(np.arange(500,Niter),fit1,label='fit_after_500')
    ax[1].loglog(np.arange(500,Niter),fit1,label='fit_after_500')
    for i in range(2):
        ax[i].legend()

for i in range(1,5):    plt.figure(i);plt.savefig('Figure_{}.png'.format(i))
#Temp
# Jmag2 = Jx**2+Jy**2 #magnitude square of Current Density
# Niter1 = Niter
# Temp = np.zeros((Ny,Nx))
# TempErrors = np.zeros(Niter1)
# for i in range(Niter1):
#     oldTemp = Temp.copy()
#     Temp[1:-1,1:-1] = (Temp[1:-1,:-2]+Temp[1:-1,2:]+Temp[:-2,1:-1]+Temp[2:,1:-
#     Temp[0],Temp[-1] = 300,Temp[-2]
#     Temp[ii] = 300
#     Temp[:,0] = Temp[:,1]
#     Temp[:,-1] = Temp[:,-2]
#     TempErrors[i] = np.max(np.abs(oldTemp-Temp))

# for i in range(1,4):    plt.close(i)
# plt.figure(5)
# plt.imshow(Temp,cmap=plt.cm.hot,interpolation='bilinear')
# fig, ax = plt.subplots(1,2,sharey = True)
# plt.title('TempError Vs N of iters in loglog and semilogy')
# ax[0].semilogy(TempErrors)
# ax[1].loglog(TempErrors)

# plt.figure(6)
# ax = p3.Axes3D(plt.gcf())
# plt.title('Surface  temperature')#plt.zlabel('potential')
# surface = ax.plot_surface(X1,Y1,Temp,rstride=1,cstride=1,cmap=plt.cm.jet)
plt.show()
```