Assignment 5 : Laplace Equation

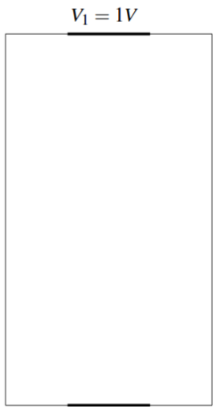
Ganga Meghanath

EE15B025

Introduction

The assignment is based on currents in a resistor. The currents depend on the shape of the resistor and we want to see if $R = \rho\frac{L}{A}$ works or not.

A voltage $V_{AB} = 1\text{V}$ is applied across the terminals of a resistor.



As a result, current flows. 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 the potential,

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

and continuity of charge yields

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

Combining these equations we obtain

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

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,

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

Code

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

def update(phi):        #Updating the potential
        phi[1:-1,1:-1]=0.25*(phi[:-2,1:-1]+\
                phi[1:-1,0:-2]+phi[2:,1:-1]+phi[1:-1,2:])
        return phi

def assert_boundaries(phi,Nbegin,Nend): #boundary conditions
        phi[1:-1,0]=phi[1:-1,1]
        phi[1:-1,-1]=phi[1:-1,-2]
        phi[0,1:Nbegin],phi[0,Nend+1:-1]=phi[1,1:Nbegin],phi[1,Nend+1:-1]
        phi[-1,1:Nbegin],phi[-1,Nend+1:-1]=phi[-2,1:Nbegin],phi[-2,Nend+1:-1]
        return phi


def fun(Nx=25,Ny=25,Nbegin=8,Nend=17,Niter=1500) :
        error=zeros(Niter)

        phi=zeros((Nx,Ny))              #create phi matrix
        phi[0,Nbegin:Nend+1]=1  #top potential 1
        phi[-1,Nbegin:Nend+1]=0 #lower potential 0

        for k in range(Niter):
                oldphi=phi.copy()
                phi=update(phi)
                phi=assert_boundaries(phi,Nbegin,Nend)
                error[k]=(abs(phi-oldphi)).max()

        logy=log(error)
        x=np.ones((len(error),2))
        x[:,1]=range(Niter)

        ans1=lstsq(x,logy)[0] #from index 0
        ans2=lstsq(x[500:,:],logy[500:])[0] #from index 500
```

```
A1,B1=exp(ans1[0]),ans1[1]
A2,B2=exp(ans2[0]),ans2[1]

q1 = B1*np.arange(Niter)
q2=B2*np.arange(Niter)

print "A and B values for error : "
print "Case 1 : {} and {}".format(A1,B1)
print "Case 2 : {} and {}".format(A2,B2)

title('Evaluation of Error with iteration')
xlabel('Iteration')
ylabel('log(Error)')
semilogy(range(Niter)[::50],error[::50],'ro',label='error')
semilogy(range(Niter)[::50],A1*np.exp(q1)[::50],'r',label='fit1')
semilogy(range(Niter)[500::50],A2*np.exp(q2)[500::50],'g',label='fit2
legend()
show()

fig1=figure(4)
ax=p3.Axes3D(fig1)
x=arange(1,Nx+1)
y=arange(1,Ny+1)
X,Y=meshgrid(x,y)
title('The 3D Surface plot of the potential')
surf=ax.plot_surface(Y,X,phi,rstride=1,cstride=1,cmap=cm.jet,linewidth
show()

title('Contour plot of potential')
contour(Y,X,phi)
show()

Jx=np.zeros(phi.shape)  #current densities
Jy=Jx.copy()

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

title('Vector plot of the current flow')
quiver(y,x,Jy.transpose(),Jx.transpose())
show()
print "Electrode between indices : {} and {}".format(Nbegin,Nend)
print "The Iavg value : {}".format((sum(Jy[:,1]) + sum(Jy[:,-2]))/2)
#Iavg
print "The Idiff value : {}".format(abs(sum(Jy[:,1]) - sum(Jy[:,-2])))
```

```
        print "Resistance : {}".format(1/((sum(Jy[:,1]) + sum(Jy[:,-2]))/2)) #
        return phi,Nx,Ny

#Order : Nx,Ny,Nbegin,Nend,Niter
fun()
fun(int(sys.argv[1]),int(sys.argv[2]),int(sys.argv[3]),\
        int(sys.argv[4]),int(sys.argv[5]))     #for fist case with small ele
phi,Nx,Ny=fun(int(sys.argv[1]),int(sys.argv[2]),0,\
        int(sys.argv[2])-1,int(sys.argv[5]))   #long electrode

array=np.zeros(phi.shape)
arr=np.linspace(Ny,Ny,Ny)-np.arange(0,Ny)
array=arr
epsilon2=sum((phi-array/Ny)**2)/(Nx*Ny)
print 'epsilon2 = {}'.format(epsilon2)
```

Output

For default arguments :

```
A and B values for error :
Case 1 : 0.00235717964439 and -0.00276031665063
Case 2 : 0.00141179338348 and -0.00226201175839
Electrode between indices : 8 and 17
The Iavg value : 0.432965780098
The Idiff value : 0.0540234032603
Resistance : 2.30965135345
```

For specified arguments :

python Assg5.py 30 30 10 19 2000

```
A and B values for error :
Case 1 : 0.00149901445721 and -0.00180659748438
Case 2 : 0.000839141749588 and -0.00138543227524
Electrode between indices : 10 and 19
The Iavg value : 0.388145592997
The Idiff value : 1.66533453694e-16
Resistance : 2.57635283781
```

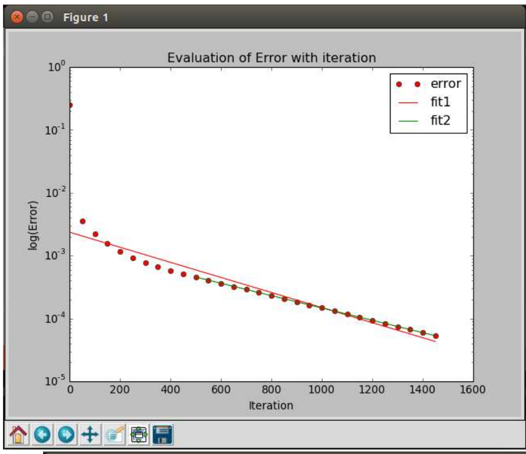
Full length electrode :

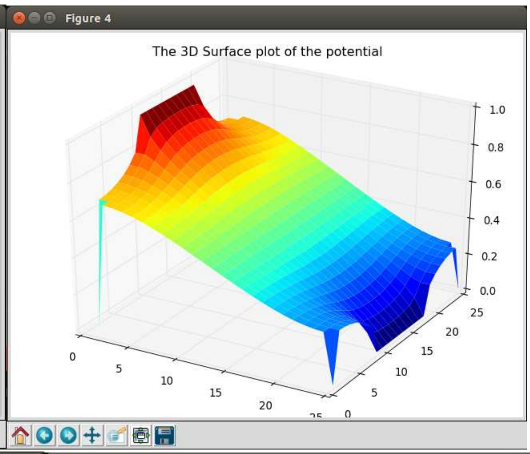
```
A and B values for error :
Case 1 : 0.00261456004012 and -0.00318576206597
Case 2 : 0.00186331495463 and -0.00293599566766
Electrode between indices : 0 and 29
The Iavg value : 0.965517241375
```

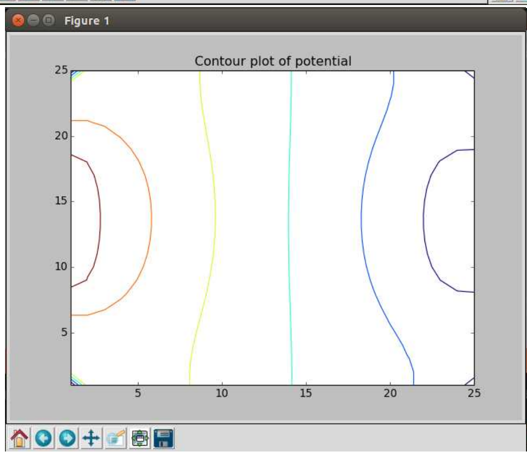
```
The Idiff value : 0.0
Resistance : 1.03571428572
epsilon2 = 0.172637302543
```

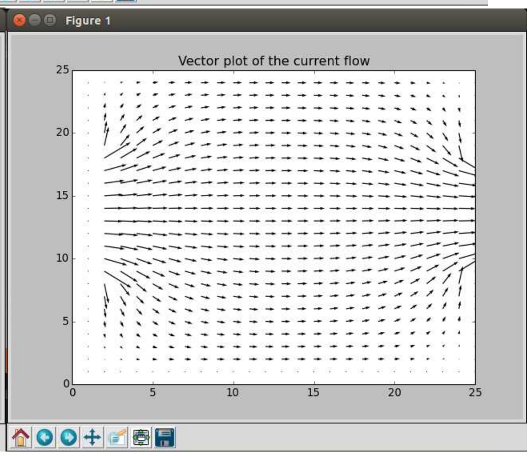
Graphs :

For default arguments :

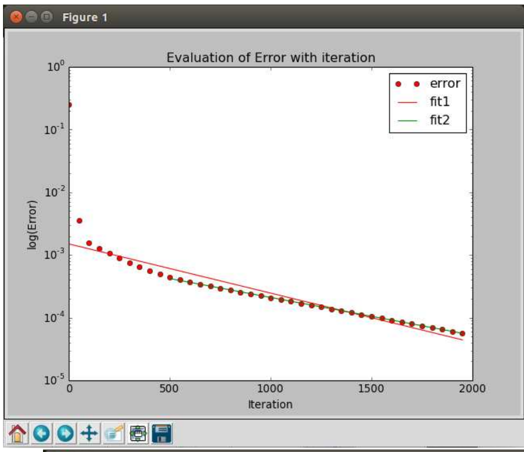


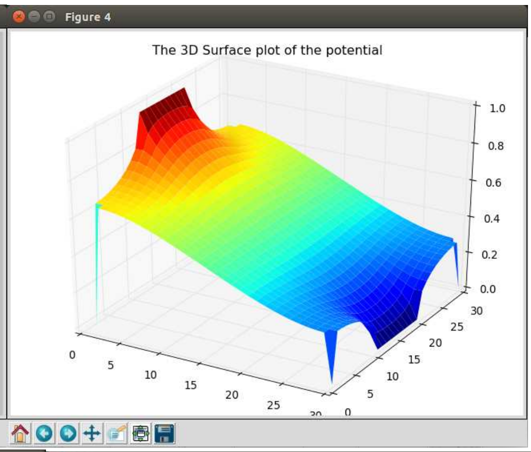


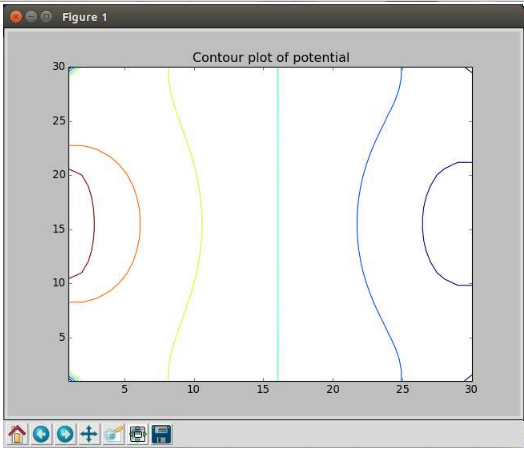


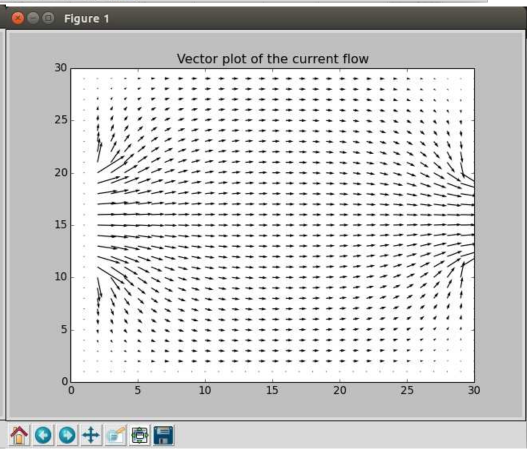


For specified arguments :

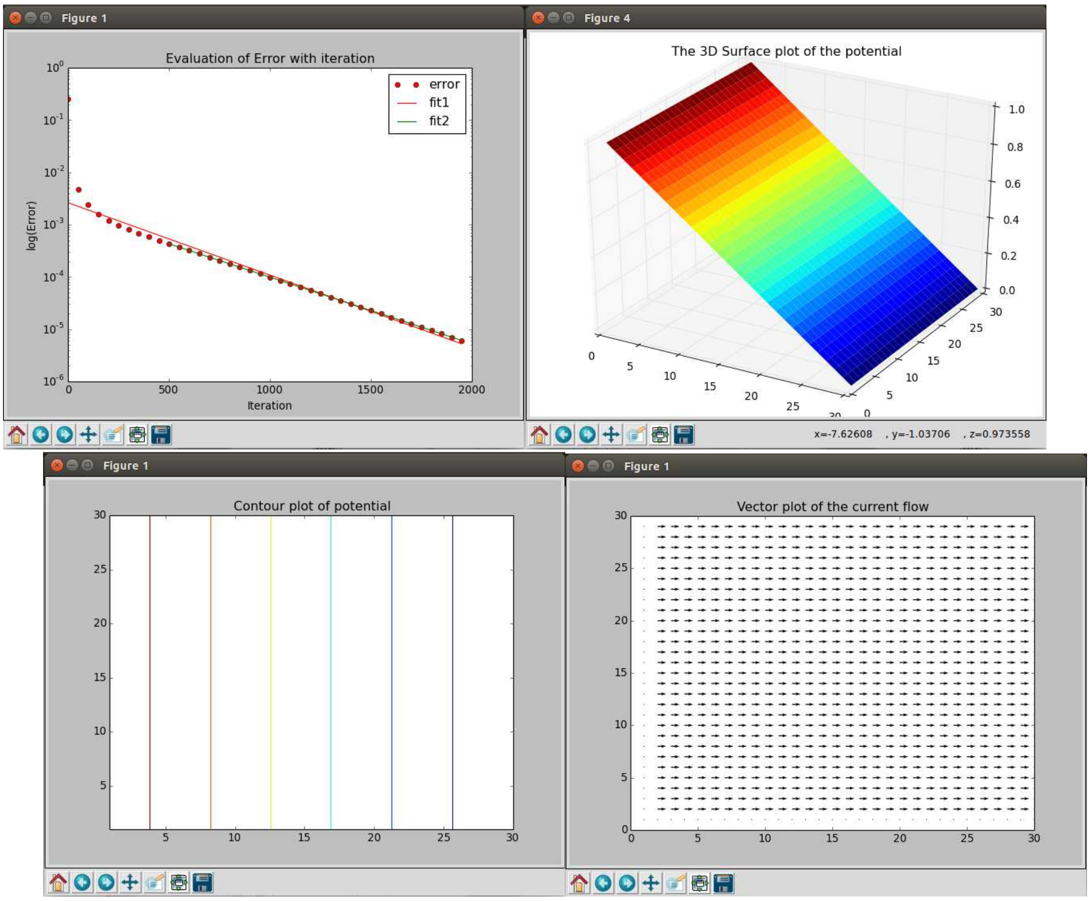








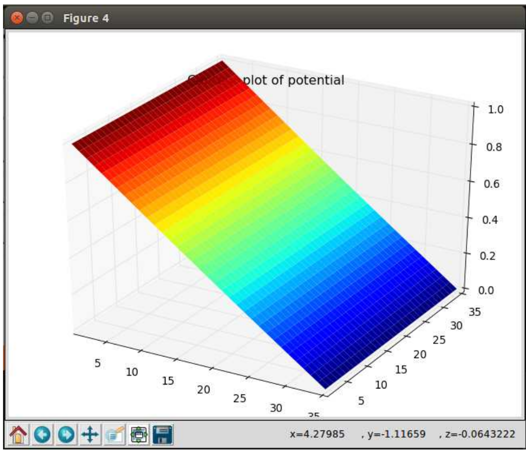
Full length electrode :

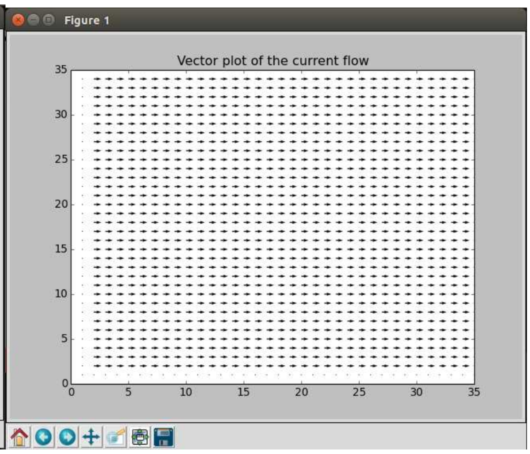


Inference

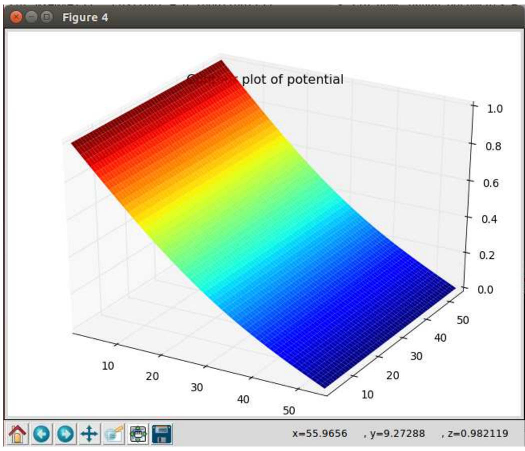
The variation of epsilon with increase in Nx and Ny can be seen below :

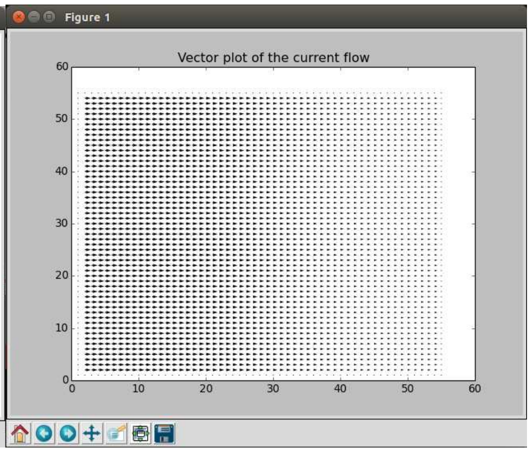
For Nx=Ny=35 , epsilon2 = 0.171900070878



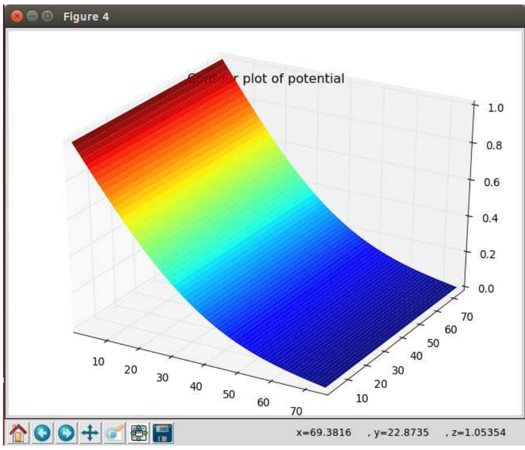


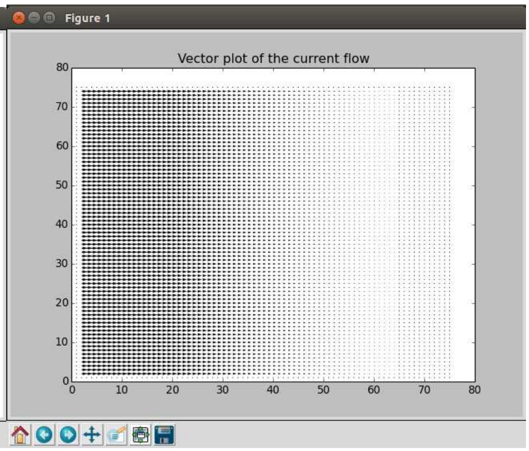
For Nx=Ny=55 , epsilon2 = 0.177761140725



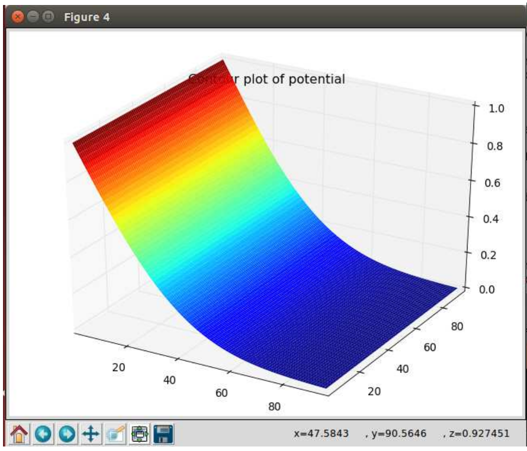


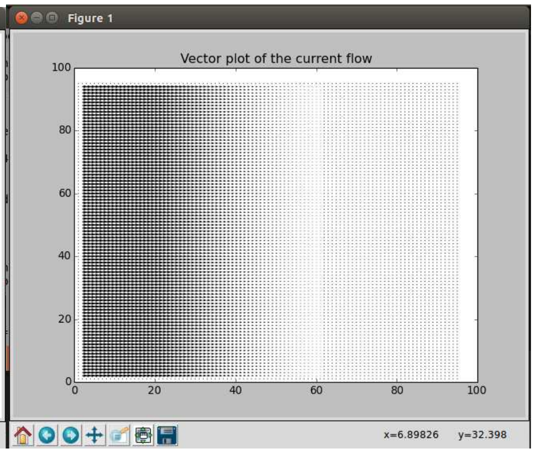
For Nx=Ny=75 , epsilon2 = 0.201395730006



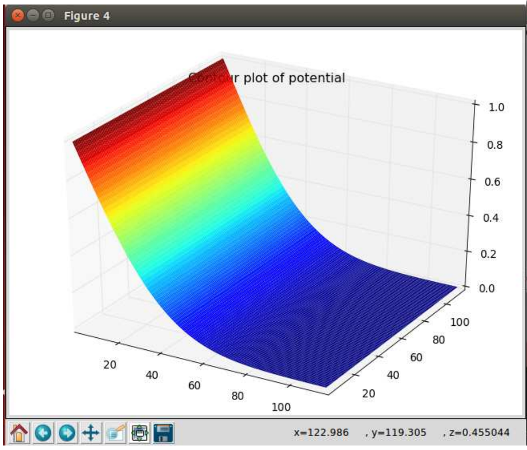


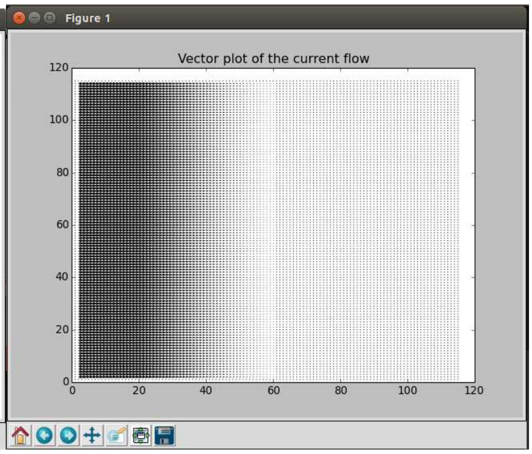
For Nx=Ny=95 , epsilon2 = 0.22632677789



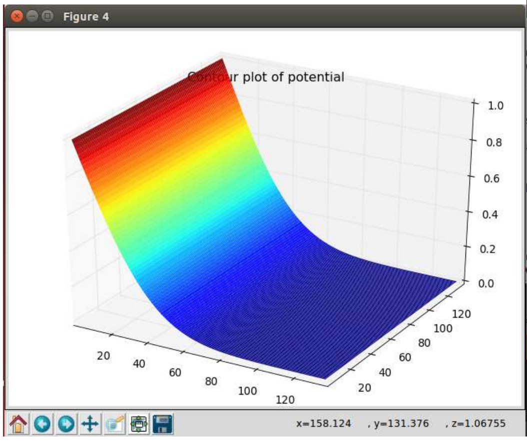


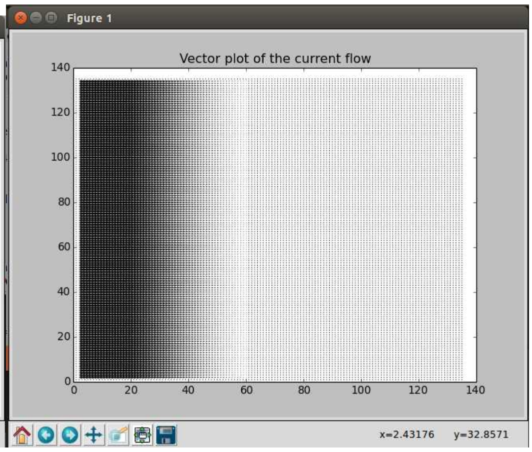
For Nx=Ny=115 , epsilon2 = 0.244922581726



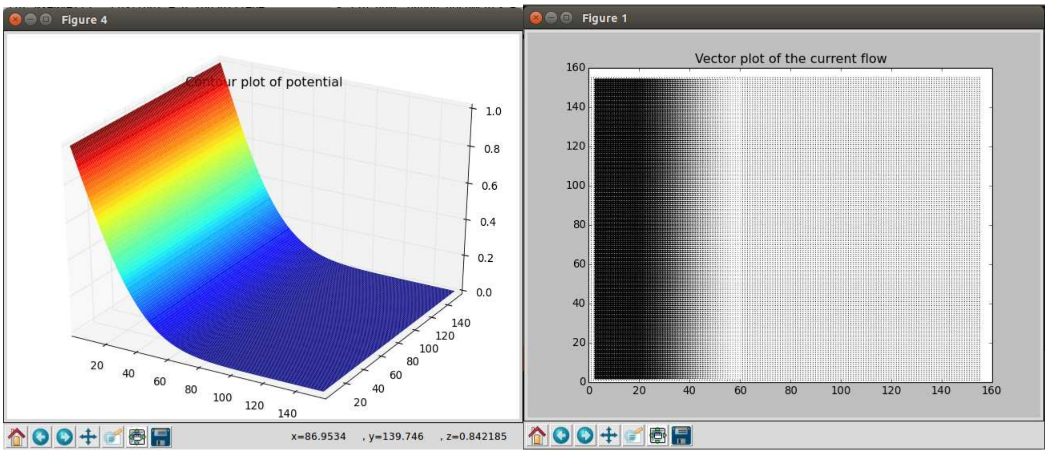


For Nx=Ny=135 , epsilon2 = 0.258230621492

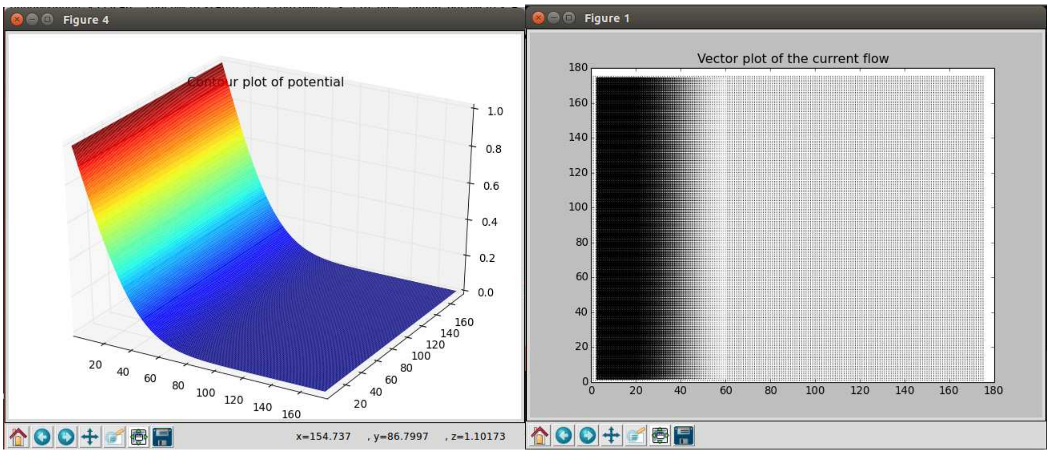




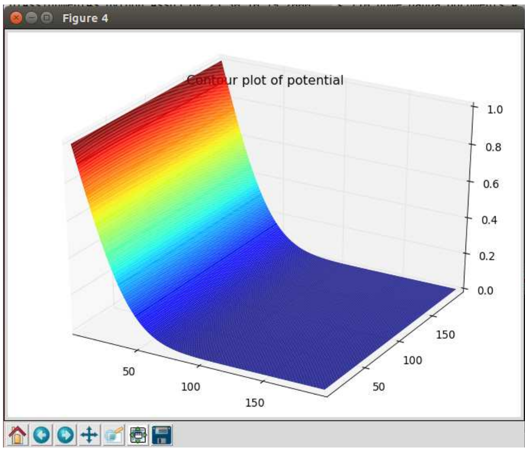
For Nx=Ny=155 , epsilon2 = 0.268077062159

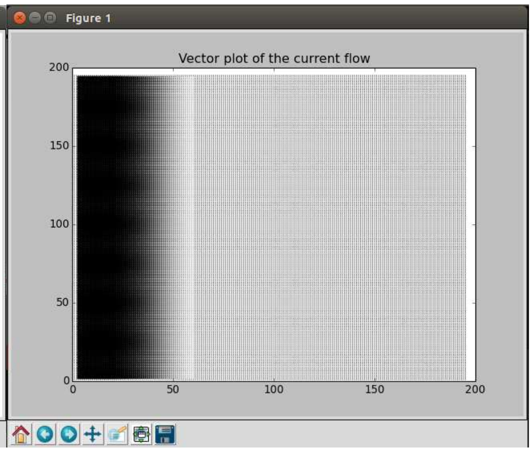


For Nx=Ny=175 , epsilon2 = 0.275642578605

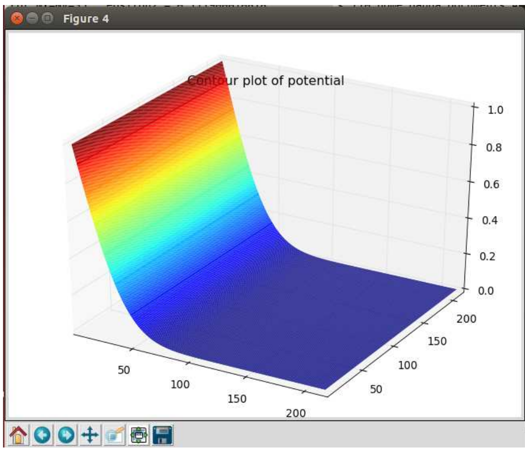


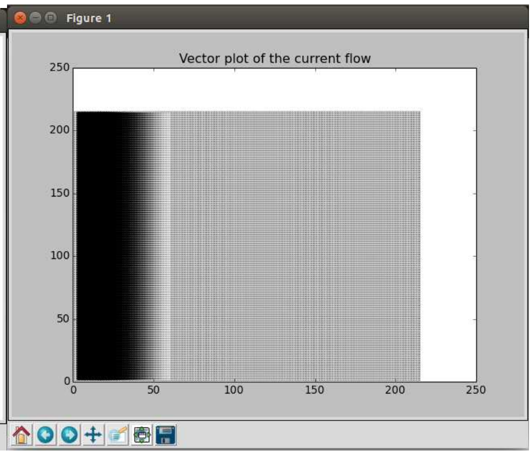
For Nx=Ny=195 , epsilon2 = 0.281636426928





For Nx=Ny=215 , epsilon2 = 0.28650217484





As we can see from the above figures, the epsilon(error) increases with Nx and Ny. This explains the change in behaviour of the figures as Nx and Ny increases. ie, the figures become more and more erroneous. This shows the ineffectiveness in using the above method for practical applications.

The code used for computing the above errors is as follows:

```
Nx=35
Ny=35
for i in range(10):
        phi,Nx,Ny=fun(Nx,Ny,0,Ny-1,2000)
        array=np.zeros(phi.shape)
        arr=np.linspace(Ny,Ny,Ny)-np.arange(0,Ny)
        array=arr
        epsilon2=sum((phi-array/Ny)**2)/(Nx*Ny)
```

```
print 'For Nx=Ny={} , epsilon2 = {}'.format(Nx,epsilon2)
Nx+=20
Ny=Nx
```

Only requires figures were displayed and rest of the code commented during the process.