EE2703 Assignment 5 : Laplace Equation

Aman Kumar EE19B066

March 27, 2021

1 The assignment

In this assignment we do the following-

- Solve for potential in a resistor.(A plate of $1cm$ x $1cm$ in this case)
- Solve for currents in that resistor.
- Learn to solve Laplace equation with an iterative approach and see how it converges.
- Learn to vectorize *for* loops

1.1 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. The current density $\vec{j}$ is related to the local Electric Field $\vec{E}$ by the conductivity σ

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

Now the Electric field is the gradient of the potential ϕ,

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

and by continuity equation

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

For DC currents and a material of constant conductivity the equation becomes the Laplace equation

$$\nabla^2 = \phi$$

Now the Laplace equation in 2 dimensions can be written in cartesian coordinates as

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

where,

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

so,

$$\frac{\partial^2\phi}{\partial x^2}_{(x_i,y_j)} = \frac{\phi(x_{i+1}, y_j) - 2\phi(x_i, y_j) + \phi(x_{i-1}, y_j)}{(\Delta x)^2}$$

Thus similarly we can do for y as well. Since $\Delta x = \Delta y$, we obtain

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

which is just the average of the potentials of its neighbouring nodes(if solution holds)

1.2 Things to do

Mainly the following things are asked in this assignment:

- Plot the contour plot of the potential before solving(iterations) the Laplace equation. Just showing the electrode nodes at 1 V potential.
- Solve the Laplace equation using iterations.
- Plot the semilog and loglog plots of Errors(max difference) in ϕ after every iteration w.r.t previous iteration.
- Plot the 3-D surface plot of the potential after solving.
- Plot a contour plot of the potential after solving(iterations) the Laplace equation.
- Plot the vector plot of currents
- Plot net error vs number of iterations.

2 Assignment questions

2.1 Defining the Parameters

The parameters that control the program has some default values. If the user correctly passes the required 4 arguments in the command line, then those values are accepted. Otherwise, default values are taken. Python code-

```
if len(argv) == 5:
    Nx = min((int(argv[1]),int(argv[2])))
    Ny = min((int(argv[1]),int(argv[2])))
    if argv[1] != argv[2]:
        print("Nx and Ny should be equal for my program")
    if int(argv[3]) > Nx/2 :
        radius = 8
        print("\nRadius given is greater than size of plate. Default value is taken.")
    else:
        radius = int(argv[3])
    if int(argv[4]) > 500:
        Niter = int(argv[4])
    else:
        Niter = 1500
        print("Niter cannot be less than 500. It's default value 1500 is taken.")
else:
    print("\nUsage: %s <Nx> <Ny> <radius> <Niter>" % argv[0])
    Nx=25
    Ny=25
    radius=8
    Niter=1500
```

2.2 Potential array(ϕ) initialization

First we initialize ϕ to zero and then find which nodes are in contact with the wire to make potential of nodes equal to 1 V. **numpy.where()** is used here to find the points inside the radius of wire:

```
        phi = np.zeros((Nx,Ny))
        x = pl.linspace(0.5,-0.5,Nx)
        y = pl.linspace(0.5,-0.5,Ny)
```

```
X,Y = pl.meshgrid(x,y)
ii = pl.where(X*X + Y*Y <= (radius/Nx)**2)
phi[ii] = 1
```

Then we plot (in Figure 1) the (filled)contour plot of the intial potential using the following code(I am omitting the non-important part here):

```
pl.contourf(X,Y,phi)
pl.title("Contour Plot of Potential(2D) before solving")
pl.scatter(X[ii],Y[ii],color="red",label="Electrode Nodes")
pl.xlabel("X($cm$)")
pl.ylabel("Y($cm$)")
pl.legend()
pl.colorbar()
pl.show()
```

2.3 Performing the Iterations

We use a second potential array, called **oldphi** to hold the values of the previous iteration. This is to keep track of how much the array changed during the current iteration. In this we have vectorized the *for loop* so that it works with almost the speed of **C**. This is the main loop of the proagram-

```
for k in range(Niter):
    oldphi = phi.copy()
    phi[1:-1,1:-1] = 0.25*(phi[1:-1,0:-2] + phi[1:-1,2:] + phi[:-2,1:-1] + phi[2:,1:-1])
    phi[ii] = 1

    #Boundary conditions.
    phi[1:-1,0] = phi[1:-1,1]                    #Top boundary
    phi[1:-1,-1] = phi[1:-1,-2]                  #Right boundary
    phi[0,1:-1] = phi[1,1:-1]                    #Left boundary
    phi[-1,1:-1] = 0                             #Lower boundary(grounded)
    errors[k] = abs(phi - oldphi).max()      #Finding the max change for this iteration
```

2.4 Plotting errors to see the convergence

The errors reduce but very slowly. The plot is linear in semilog. In loglog it is reasonably linear upto first 500 iterations, after that it decays exponentially. We plot every 50^{th} point. We fit the error to the form

$$y = Ae^{Bx}$$

We extract the above fit for the entire vector of errors and for those error entries after the 500^{th} iteration. The plot of actual errors, errors using fit1 and fit2 are plotted in Figure 2 and Figure 3 for *semilogy* and *loglog* respectively.

2.5 Surface plot of potential

For plotting a 3-D surface plot(Figure 4) of ϕ the following code is used:

```
fig3 = pl.figure(3)
ax=p3.Axes3D(fig3)
pl.title("The 3-D surface plot of the potential")
surf = ax.plot_surface(X,Y, phi, rstride=1, cstride=1, cmap=pl.cm.jet,linewidth=1)
pl.xlabel("$x(cm)$")
pl.ylabel("$y(cm)$")
```

```
pl.show()
```

2.6 Contour Plot of the Potential

This contour plot(Figure 5) is after the iteations are done. i.e. The solution for potential in the resistive plate. Python code:

```
fig4 = pl.figure(4)
pl.contourf(X,Y,phi)
pl.title("Contour Plot of Potential(2D) after solving")
pl.scatter(X[ii],Y[ii],color="red",label="Electrode Nodes")
pl.xlabel("X($cm$)")
pl.ylabel("Y($cm$)")
pl.legend()
pl.colorbar()
pl.show()
```

2.7 Vector Plot of Currents

For obtaining the currents we need to compute gradient of potential ϕ. Since the actual value of conductivity σ does not matter to the shape of the current profile, so we set it to unity.

$$j_x = -\frac{\partial \phi}{\partial x}$$

$$j_y = -\frac{\partial \phi}{\partial y}$$

Numerically,

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

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

$\vec{J}$ is plotted using *pylab.quiver()* in Figure 6 as follows:

```
#Arrays to store the x and y component of currents
Jx = np.zeros((Nx,Ny))
Jy = np.zeros((Nx,Ny))

#Calculating vector current density components
Jx = -0.5*(phi[1:-1,:-2] - phi[1:-1,2:])
Jy = -0.5*(phi[:-2,1:-1] - phi[2:,1:-1])

#Plotting the vector Plot of Currents
fig5 = pl.figure(5)
pl.quiver(X[1:-1,1:-1],Y[1:-1,1:-1],Jx,Jy)
pl.scatter(X[ii],Y[ii],color="red")
pl.title("The vector plot of the current flow")
pl.xlabel("X($cm$)")
pl.ylabel("Y($cm$)")
pl.show()
```

2.8 Net error vs number of iterations

The maximum error scales as

$$error_k = Ae^{Bx}$$

where k is the iteration number. For the default case, I got

$$error_k = 0.021525 \exp(-0.01295k)$$

Summing up the errors, we have

$$netError < -\frac{A}{B} exp(B(N + 0.5))$$

Thus, using this relation and our estimated values of A and B we plot(Figure 7) netError vs N(number of iterations). From $N = 500$ to $N = 2Niter$. Python code for doing that:

```
def net_error(A,B):
    return ((-A/B)*np.exp(B*(iter_ + 0.5)))

NetError = net_error(A,B)
fig6 = pl.figure(6)
pl.semilogy(iter_,NetError,'ro',linestyle='dashed',linewidth=1,label='Net Error')
pl.title("Net Error vs Number of iterations")
pl.xlabel("Niter")
pl.ylabel("Net errror")
pl.legend()
pl.grid(True)
pl.show()
```

3 Plots and Results

3.1 Initial potential

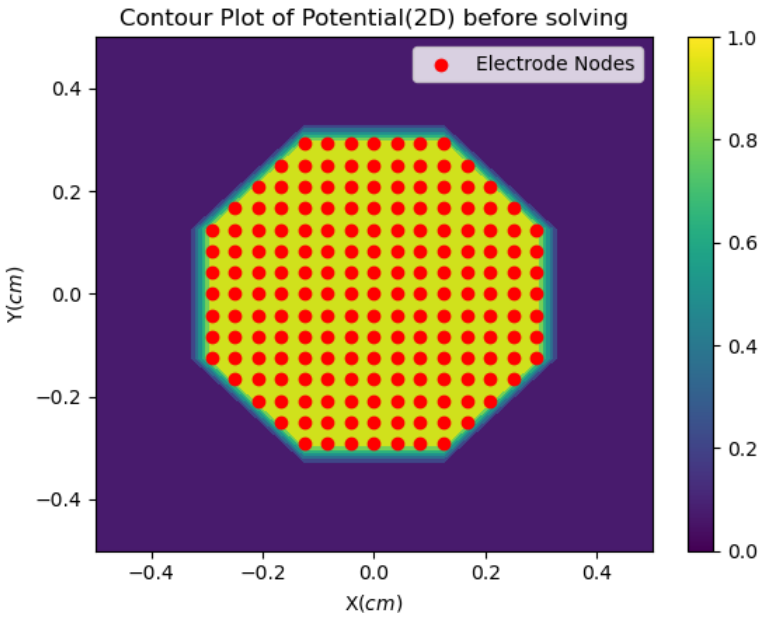

Figure 1: Initial potential

This plot is before solving the Laplace equation. Red dots denote the nodes in contact with the electrode.They are at 1 V.

3.2 Errors

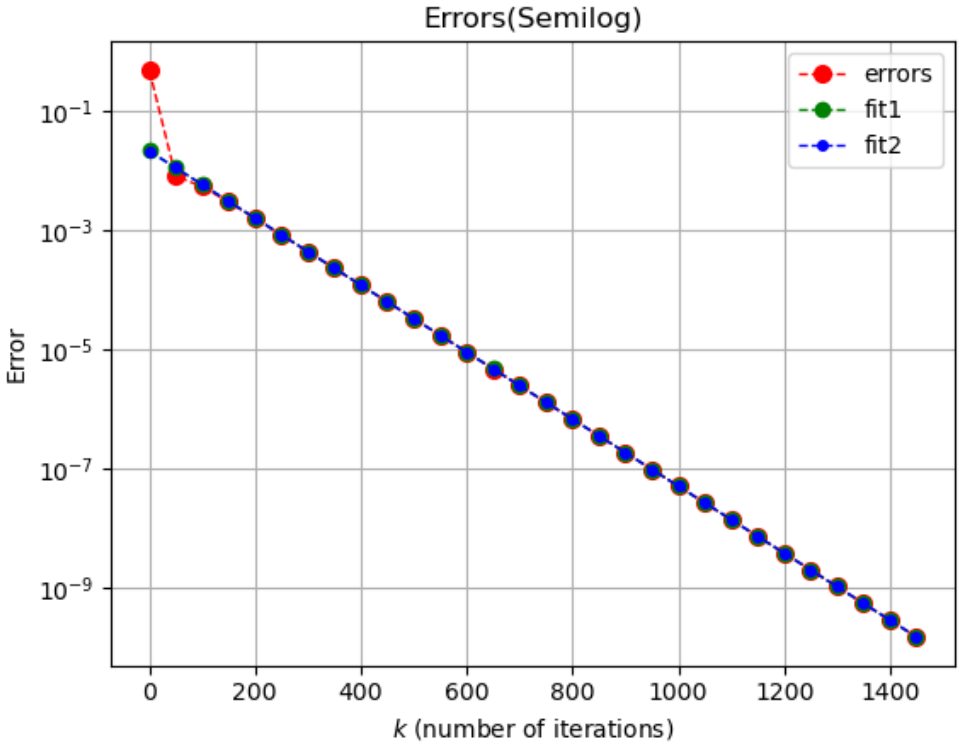

Figure 2: Errors on semilog

errors are the actual errors, while *fit1* is the fit that we got by taking the entire error vector, and fit 2 is what we got from taking error entries after the 500^{th} point. We can see that error is linear on

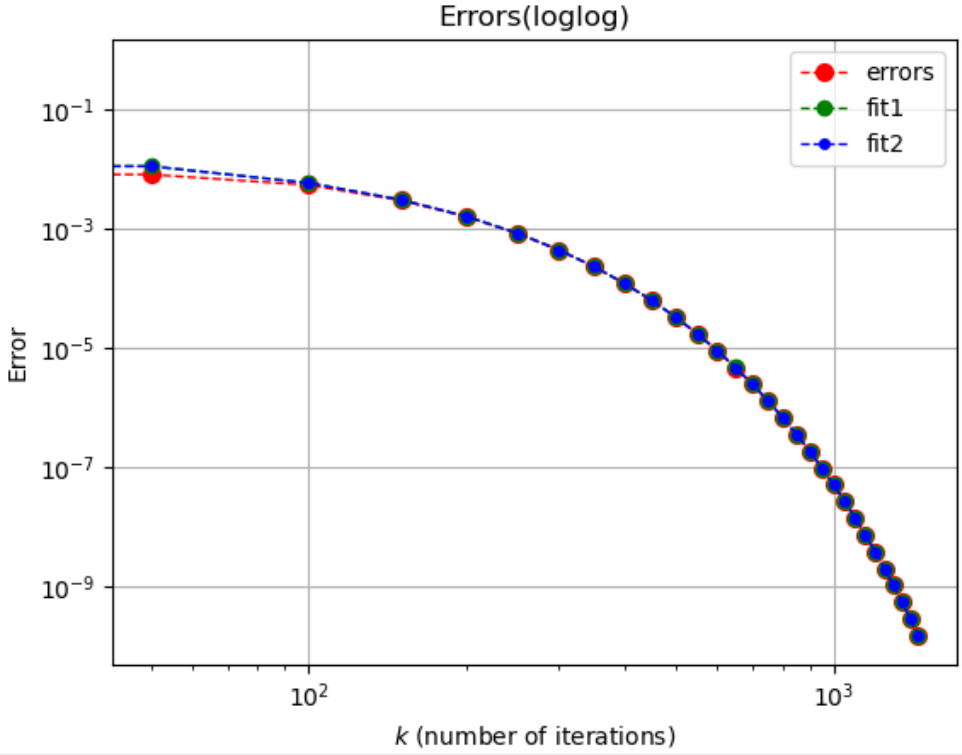

Figure 3: Errors on loglog

semilog scale. In loglog scale it is linear initially but then it enters the exponential regime.

3.3 Potential after solving

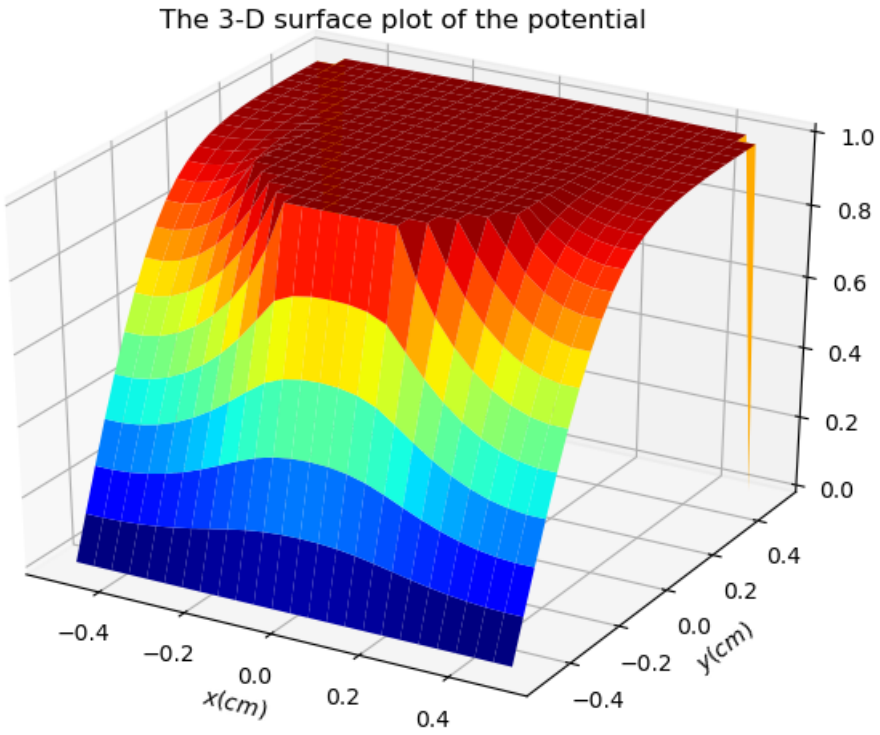

Figure 4: 3-D surface plot of potential

We can see the from the plot that potential is higher on the top of the plate and reduces to zero till bottom. Also since the electrode is connected to the centre of the plate, it is 1 V in the central region within the radius. Thus we expect that the current will flow from the central region to the bottom of the resistor. It will disperse a little in the middle.

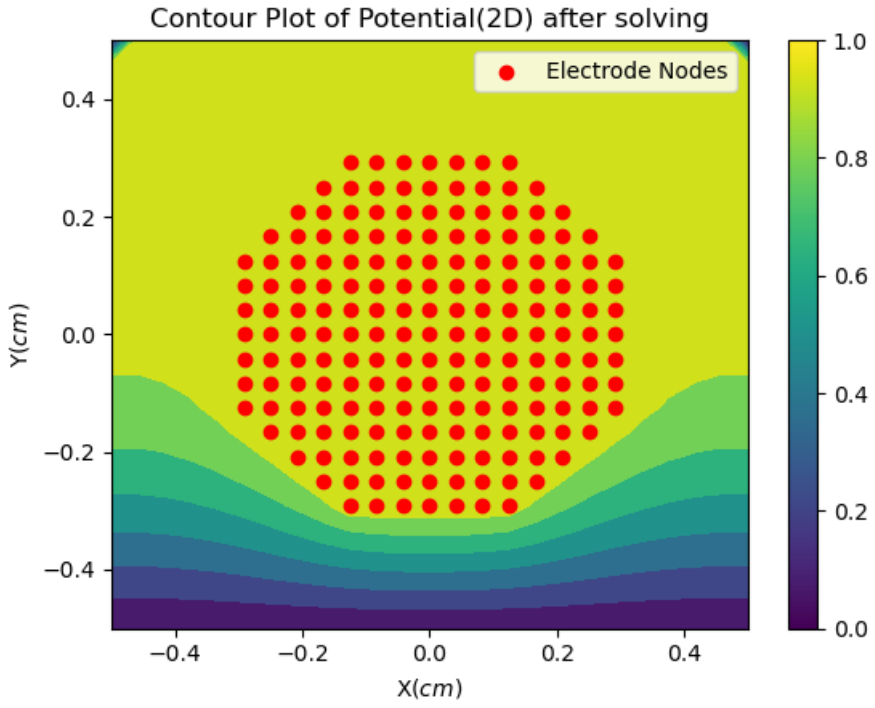

Figure 5: Contour plot of Plotential

3.4 Currents

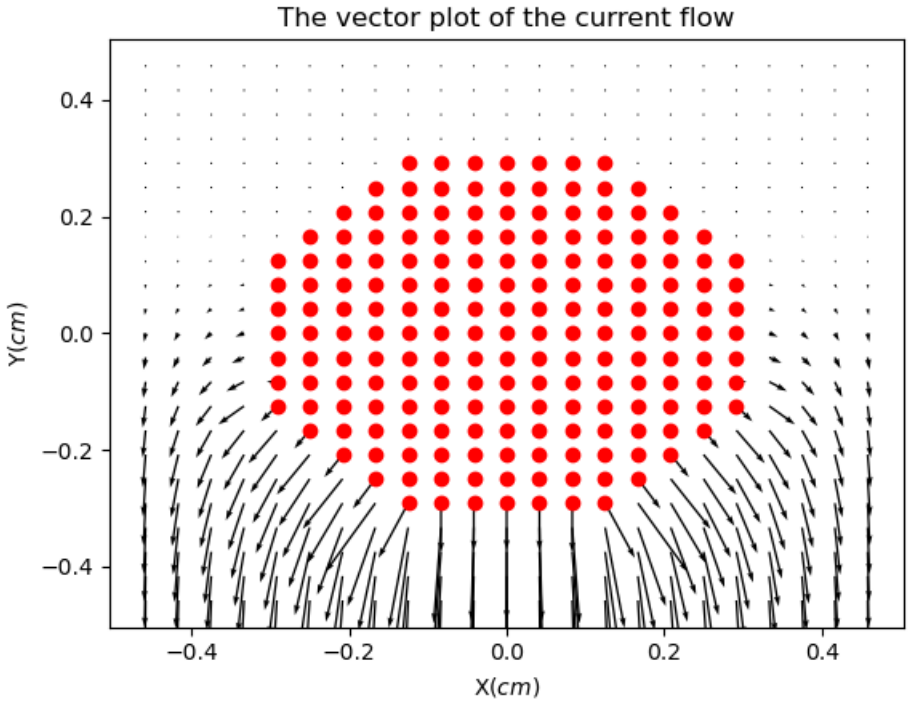

Figure 6: Vector plot of currents

Thus as expected, current flows from the central region to the bottom of the resistor. It disperses a little in the middle. The part of the current directly below the central region carries major part of the current. As a result heating will be more in this region. While the top edge won't heat much.

3.5 Net error vs Number of iterations

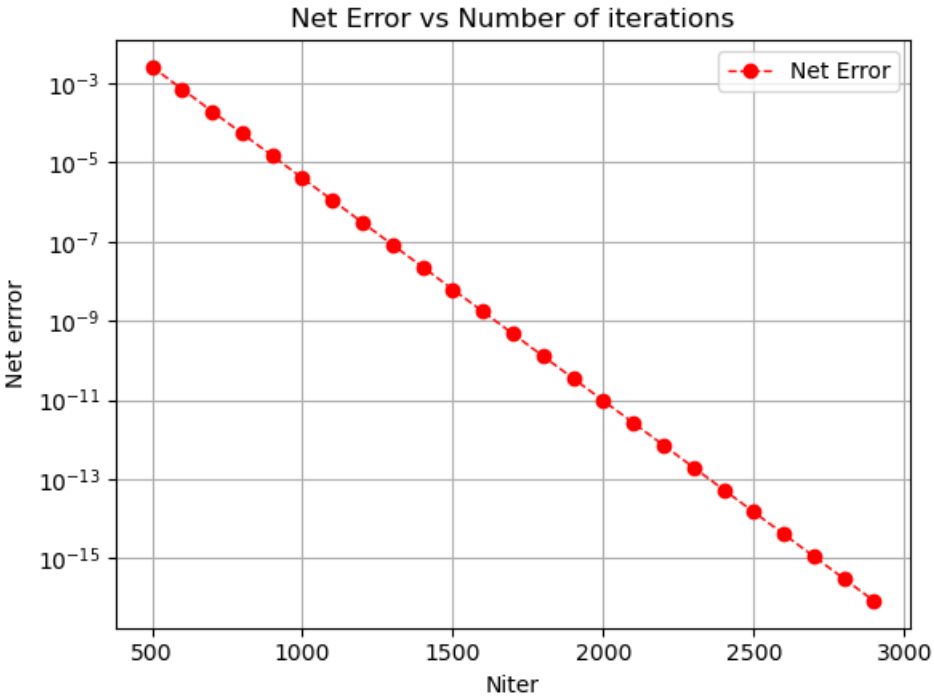

Figure 7: Net error vs iterations

The net error reduces with the increase in number of iterations as expected. The semilog plot is linear which means that as the number of iterations increase the decrease in error becomes slower and slower.

4 Conclusions

- A mojor part of the current flows only in the narrow region at the bottom.
- Almost no current flows in the top part of the resistive plate.
- To reduce the error, we have to given large values of *Niter*, *Nx* and *Ny* to get results with better accuracy.
- This method of solving Laplace equations is very very slow as even after vectorizing almost all the *for* loops, the program is a bit slow.