APPLIED PROGRAMMING LAB
(Week - 3)
Estimation of Potential and Current for a given Electrode Configuration

Shubhangi Ghosh
EE15B129
Department of Electrical Engineering

February 23, 2017

1 INTRODUCTION:

This report presents:

1. Computation of potential by solving Poisson's equation in two dimensions.

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = 0 \tag{1}$$

2. Computation of current by effectively differentiating potential.
3. Estimating total error by computing error between successive iterations.

1.1 Potential Computation:

The function '**potential_func(elec_size, Nx, Ny):**' is used for this purpose.

1.1.1 Potential for Non-Boundary points of Conductor

Poisson's equation in two dimensions is written as in Eqn. (1).
Assuming ϕ is known at position (x_i, y_j) in the conductor, the Possion's equation can be differentiated as:

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

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

Similarly double differentiating $w.r.t.\ y$, we get,

$$\frac{\partial^2\phi}{\partial y^2}_{(x_i,y_j)} = \frac{\phi(x_i, y_{j+1}) - 2\phi(x_i, y_j) + \phi(x_i, y_{j-1})}{(\triangle x)^2} \tag{3}$$

Substituting (2) and (3) in (1),we get:

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

So, we see that potential at a point is the average of its surrounding points, unless the point is at the boundary.
In code, this is implemented as:

```
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])
```

1.1.2 Potential at Boundary Points:

We draw boundary conditions based on the fact that electic field inside a conductor is 0.
Thus, the potential at boundary points except for those which represent the electrode is given by the potential of the non-boundary point one point off from it.

The code for this is:

```
        phi[1:-1,0]=phi[1:-1,1]
phi[1:-1,-1]=phi[1:-1,-2]
phi[0,:Nbegin]=phi[1,:Nbegin]
phi[0,(Nend+1):]=phi[1,(Nend+1):]
phi[-1,:Nbegin]=phi[-2,:Nbegin]
phi[-1,(Nend+1):]=phi[-2,(Nend+1):]
```

1.2 Current Computation:

The current density in the conductor along $x-$ or $y-$axis is computed by differentiating

$$J_x = -\frac{\partial \phi}{\partial x} \tag{5}$$

When represented in discrete domain,

$$J_{x,ij} = \frac{1}{2}(\phi_{i,j-1} - \phi_{i,j+1}) \tag{6}$$

Analogous to (5)and (6), along y-axis,

$$J_y = -\frac{\partial \phi}{\partial y} \tag{7}$$

$$J_{y,ij} = \frac{1}{2}(\phi_{i-1,j} - \phi_{i+1,j}) \tag{8}$$

In the function, 'error_est(errors,Nx,Ny):', the above is implemented in code as:

```
    Jy[1:-1,1:-1]=0.5*(phi[0:-2,1:-1] - phi[2:,1:-1])
Jx[1:-1,1:-1]=0.5*(phi[1:-1,0:-2] - phi[1:-1,2:])
```

1.3 Error Estimation:

Curve Fitting of Error between successive iterations:

We model the error between two successive iterations of potential computation as:

$$y = Ae^{Bx} \tag{9}$$

Taking *log* on both sides of (9),we get

$$log(y) = logA + Bx \tag{10}$$

We obtain A and B by least square estimation:

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ ... & ... \\ 1 & x_{N-1} \\ 1 & x_N \end{pmatrix} \begin{pmatrix} logA \\ B \end{pmatrix} = \begin{pmatrix} log(y_1) \\ log(y_2) \\ ... \\ log(y_N) \end{pmatrix} \tag{11}$$

where,

$$x_i - Iteration\, number$$

$$y_i - error\, between\, i^{th}\, and\, (i-1)^{st}\, iteration$$

In code, this is implemented in the function '**error_est(errors,Nx,Ny):**' as:

```
    #calculating B for exponential fit
a = ones((len(errors),2)) # allocate a matrix A with ones
#first column is ones
a[:,1] = iters #second column is iteration number
b = np.log(errors)#result side is log of errors
lst_sq = np.linalg.lstsq(a,b)[0] #least square approximation
#A and B are found
A = np.exp(lst_sq[0])
B = lst_sq[1]
#curve fit for errors = y found using A and B
y = A*np.exp(B*array(iters))
```

For a better curve-fit, instead of imposing the exponential fit from the first iteration, we impose the fit from the 500^{th}observation because by then the error has considerably reduced and further reduction in error fits the exponential form better.

In the code for the given experiment, we estimate error in three different situations:

1.3.1 Error from value at completed iterations to infinite iterations:

In this method, we sum up the errors between successive iterations from the currently computed iteration till infinite iterations.

$$\begin{aligned} error &= \sum_{k=N+1}^{\infty} Ae^{Bk} \\ &\approx \int_{N+0.5}^{\infty} Ae^{Bk}dk \\ error &= -\frac{A}{B}e^{(B(N+0.5))} \end{aligned} \tag{12}$$

In code, this is implemented as:

```
#expression for estimated error between potentials
print "Final estimated error in potential: %f" %((-A/B)*np.exp(B*(2000.0+0.5)))
```

1.3.2 Error of estimated potential from actual potential

This method of potential estimation is used when the electrode covers the entire edge, and thereby it is simplistic to compute potential through Poisson's equation mathematically.

$$\phi = V_0 \frac{L-y}{L}$$

Computing Mean-Square Error,

$$\epsilon_2 = \frac{1}{N_x N_y} \sum_{ij} (\phi_{ij} - V_0 \frac{L-y}{L})^2 \tag{13}$$

In code, this is done as:

```
#Mean square error in potential
def meansq_err(phi,Nx,Ny):
idx = zeros((Ny,Nx)) #initialising index matrix
#creating a matrix which contains the y-index for phi marix
idx[0:,]=np.array(range(Ny))
idx = idx.transpose()
idx=array(idx)
phi=array(phi)
#average least square error by finding sum of squares of errors
#and dividing by no. of points at which potential is estimated(Nx*Ny)
meansq_err = (1.0/(Nx*Ny))*sum((phi-1.0*((Ny-idx)/Ny))**2)
return meansq_err
```

1.3.3 Error between entering and exiting currents

There is a non-zero difference between the current entering and exiting the conductor, because the estimation doesn't represent the exact potential in a conductor.

In code, this is implemented as:

```
    #finding total current
curr_enter = sum(Jy[1,1:-1]) #adding current density in y-direction at entry
curr_exit = sum(Jy[-2,1:-1]) #adding current density in y-direction at exit
curr_avj = (curr_exit+curr_enter)/2.0 #average current
curr_diff = fabs(curr_enter-curr_exit) #diffenence or error in current
print "Niter: %d Iavj=%f Idiff=%f" %(Niter, curr_avj, curr_diff)
```

However, this error is much larger than the error as estimated in subsection **1.3.1** probably because the error at multiple points(the whole edge) gets added up.

1.4 Finding Resistance:

Resistance is calculated by dividing the elecrode voltage, V_0 by average current, $curr\_avj$.

```
    #finding resistance
print "The resistance is thus approximated to be: %f ohms." %(V0/curr_avj)
```

2 CODE:

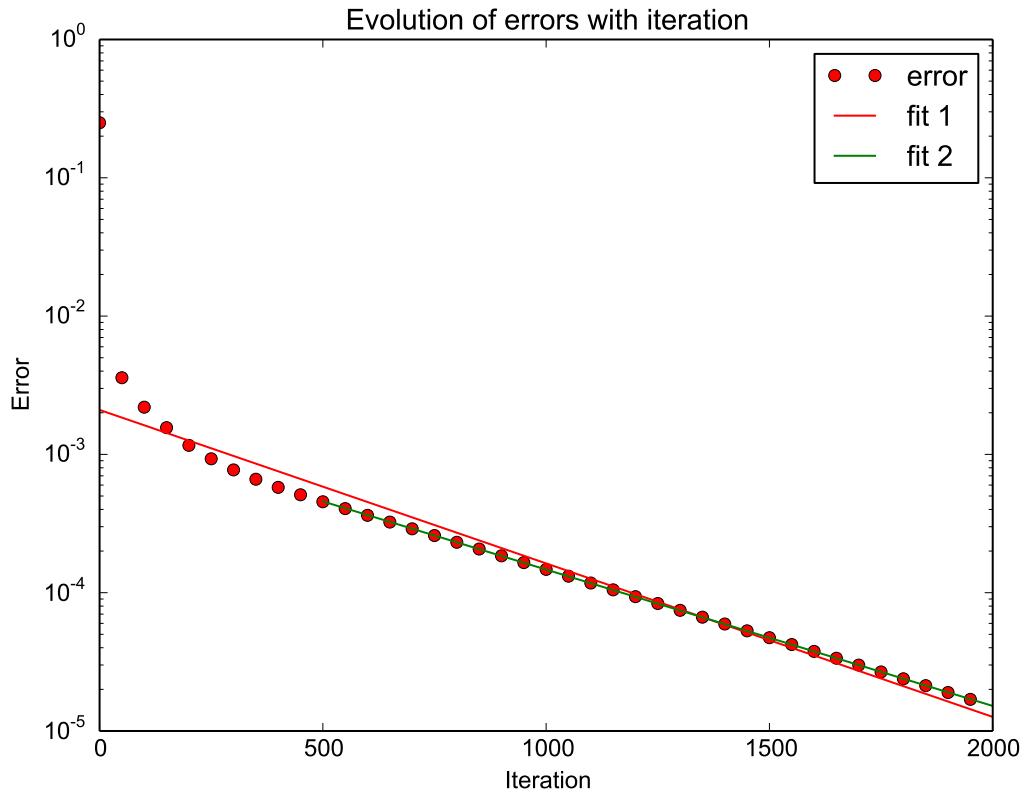
```
#Importing header files
import numpy as np
from scipy import *
from matplotlib.pyplot import *
from scipy.integrate import quad
from math import *
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
import sys
if (len(sys.argv)>1):
elec_size=sys.argv[1] #electrode size
Nx=sys.argv[2] #size along x
Ny=sys.argv[3] #size along y
Niter=sys.argv[4] #number of iterations to perform
else:
Nx=25 #size along x
Ny=25 #size along y
elec_size=10 #electrode size
Niter=2000 #number of iterations to perform
#function to calculate potential
def potential_func(elec_size, Nx, Ny):
#electrode assumed to be in the middle
Nbegin = int((Nx-elec_size)/2)+1 #starting point of electrode
Nend = Nbegin+ elec_size-1 #ending point of electrode
phi = zeros((Ny,Nx)) #intialising potential matrix
V0=1.0 #voltage of top electrode
phi[0,Nbegin:(Nend+1)]=V0
#calcutating error between two iteration
errors = []
iters = range(Niter)
for k in iters:
#copying phi
oldphi = phi.copy()
#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])
#boundary conditions
#at boundary electric field is 0, so value of phi at one edge off boundary is same as b
phi[1:-1,0]=phi[1:-1,1]
phi[1:-1,-1]=phi[1:-1,-2]
phi[0,:Nbegin]=phi[1,:Nbegin]
phi[0,(Nend+1):]=phi[1,(Nend+1):]
phi[-1,:Nbegin]=phi[-2,:Nbegin]
phi[-1,(Nend+1):]=phi[-2,(Nend+1):]
errors.append(fabs(phi-oldphi).max())
return phi, errors
#error estimation
def error_est(errors,Nx,Ny):
V0=1.0
iters = range(Niter)
title('Evolution of errors with iteration')
xlabel('Iteration')
```

```
ylabel('Error')
semilogy(iters[::50],errors[::50],'ro')
#calculating B for exponential fit
a = ones((len(errors),2)) # allocate a matrix A with ones
#first column is ones
a[:,1] = iters #second column is iteration number
b = np.log(errors)#result side is log of errors
lst_sq = np.linalg.lstsq(a,b)[0] #least square approximation
#A and B are found
A = np.exp(lst_sq[0])
B = lst_sq[1]
#curve fit for errors = y found using A and B
y = A*np.exp(B*array(iters))
semilogy(iters,y,'r')
#calculating B for exponential fit starting from 500th iteration for better fit
a = ones((len(errors)-500,2)) # allocate a matrix A with ones
#first column is ones
a[:,1] = iters[500:] #second column is iteration number
b = np.log(errors[500:]) #result side is log of errors
lst_sq = np.linalg.lstsq(a,b)[0]#least square approximation
#A and B are found
A = np.exp(lst_sq[0])
B = lst_sq[1]
print "Least square approximation: A=%f, B=%f" %(A, B)
#curve fit for errors = y found using A and B
y = A*np.exp(B*array(iters[500:]))
semilogy(iters[500:],y,'g')
legend(('error','fit 1','fit 2'))
savefig('err_fit'+str(elec_size)+'.pdf',format='pdf')
show()
#expression for estimated error between potentials
print "Final estimated error in potential: %f" %((-A/B)*np.exp(B*(2000.0+0.5)))
#Surface plot for potential
fig1=figure(2) # open a new figure
ax=p3.Axes3D(fig1) # Axes3D is the means to do a surface plot
x=arange(1,Nx+1) # create x and y axes
y=arange(1,Ny+1)
X,Y=meshgrid(x,y) # creates arrays out of x and y
title('The 3-D surface plot of the potential')
surf = ax.plot_surface(Y, X, phi, rstride=1, cstride=1, cmap=cm.jet,linewidth=0)
savefig('surface_plot'+str(elec_size)+'.pdf',format='pdf')
show()
#Contour plot for potential
fig2=figure(3)
CS = contour(x, y, phi)
clabel(CS, inline=1, fontsize=10)
title('Contour plot for potential')
savefig('contour_plot'+str(elec_size)+'.pdf',format='pdf')
show()
#current  density
Jx=zeros((Ny,Nx))
Jy=zeros((Ny,Nx))
#calculating current density for inner matrix points by effectively differentiating pot
```

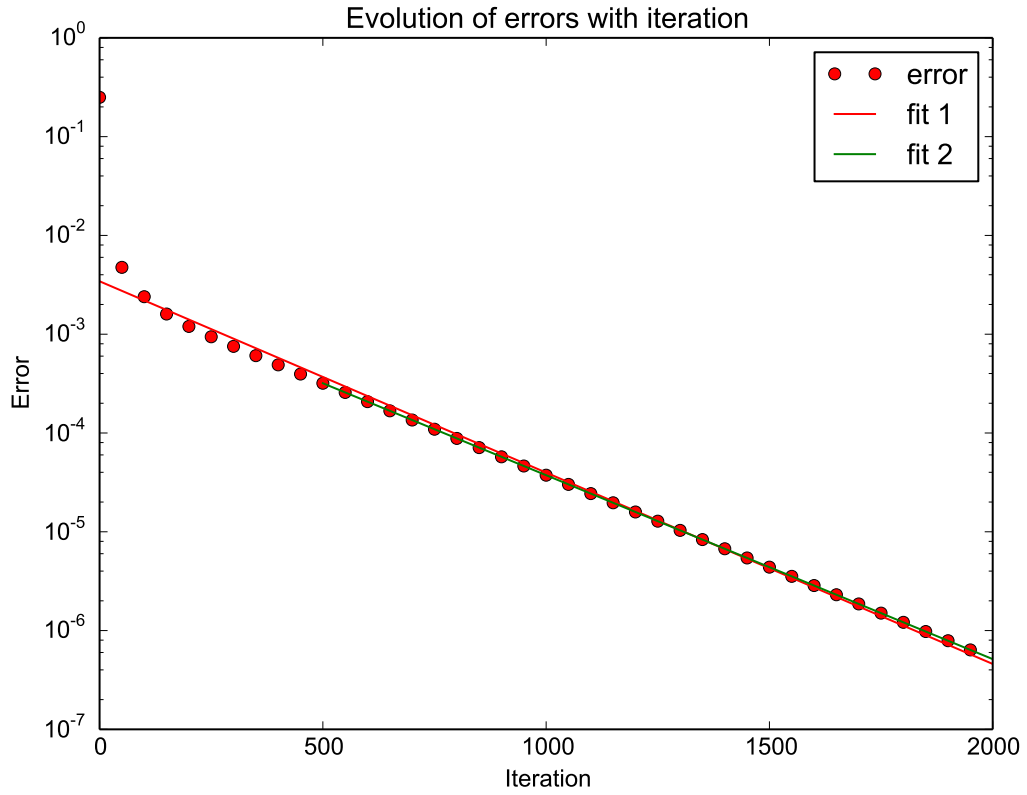
```
Jy[1:-1,1:-1]=0.5*(phi[0:-2,1:-1] - phi[2:,1:-1])
Jx[1:-1,1:-1]=0.5*(phi[1:-1,0:-2] - phi[1:-1,2:])
title('The vector plot of current flow')
quiver(y,x,Jy.transpose(),Jx.transpose())
savefig('curr_plot'+str(elec_size)+'.pdf',format='pdf')
show()
#finding total current
curr_enter = sum(Jy[1,1:-1]) #adding current density in y-direction at entry
curr_exit = sum(Jy[-2,1:-1]) #adding current density in y-direction at exit
curr_avj = (curr_exit+curr_enter)/2.0 #average current
curr_diff = fabs(curr_enter-curr_exit) #diffenence or error in current
print "Niter: %d Iavj=%f Idiff=%f" %(Niter, curr_avj, curr_diff)
#finding resistance
print "The resistance is thus approximated to be: %f ohms." %(V0/curr_avj)
#Mean square error in potential
def meansq_err(phi,Nx,Ny):
idx = zeros((Ny,Nx)) #initialising index matrix
#creating a matrix which contains the y-index for phi marix
idx[0:,]=np.array(range(Ny))
idx = idx.transpose()
idx=array(idx)
phi=array(phi)
#average least square error by finding sum of squares of errors
#and dividing by no. of points at which potential is estimated(Nx*Ny)
meansq_err = (1.0/(Nx*Ny))*sum((phi-1.0*((Ny-idx)/Ny))**2)
return meansq_err
phi,errors=potential_func(elec_size,Nx,Ny)
error_est(errors,Nx,Ny)
phi,errors=potential_func(Nx,Nx,Ny) #elec_size = Nx
error_est(errors,Nx,Ny)
#comparing least square errors for diff values of Nx and Ny while maintaining their rat
lstsq_err(phi,Nx,Ny)
Nx=50
Ny=50
phi=potential_func(Nx,Nx,Ny)[0]
lstsq_err(phi,Nx,Ny)
Nx=100
Ny=100
phi=potential_func(Nx,Nx,Ny)[0]
error_lstsq =  lstsq_err(phi,Nx,Ny)
#plotting mean square error for different Nx and Ny
print "Mean square error for Nx=%d,Ny=%d and no. of iterations = %d is %f" %(Nx,Ny,Nite
iters = range(Niter)
title('Evolution of errors with No. of point of estimation')
xlabel('Iteration')
ylabel('Error')
rng = range(10,51)
err=[]
for i in rng:
err.append(meansq_err(potential_func(i,i,i)[0],i,i))
plot(rng,err)
savefig('err_plot.pdf',format='pdf')
show()
```

3 PLOTS:

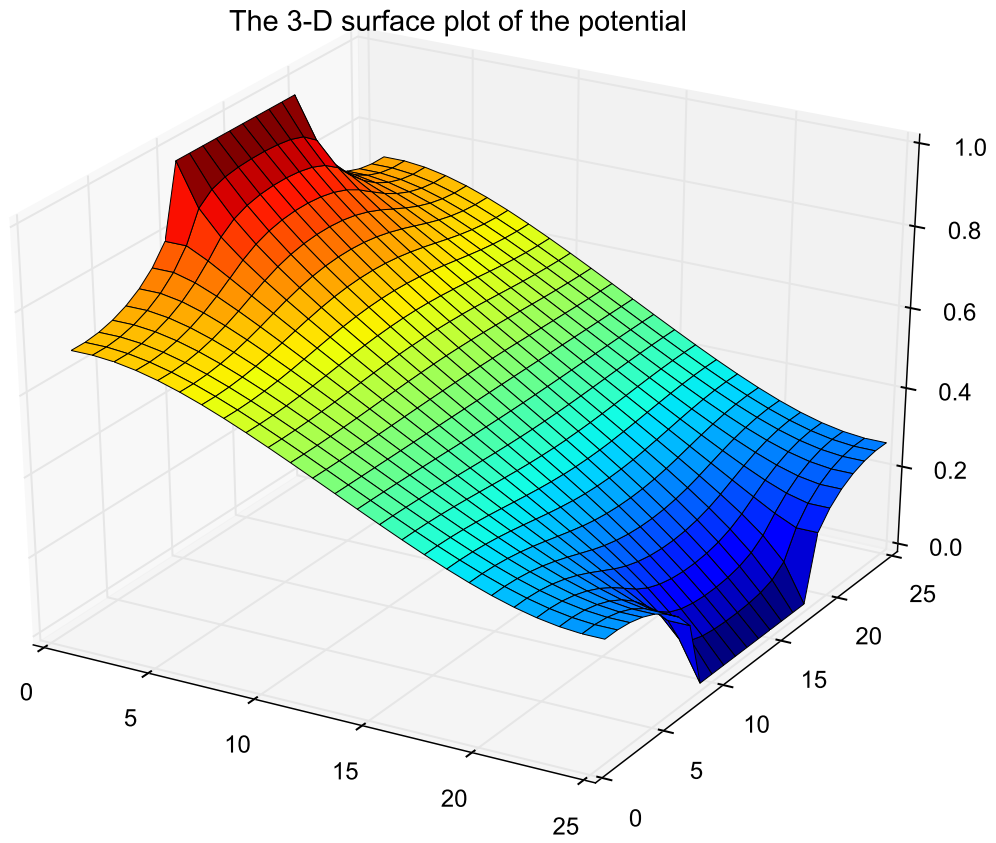
3.0.1 Error Curve Fitting for Electrode Length = 10



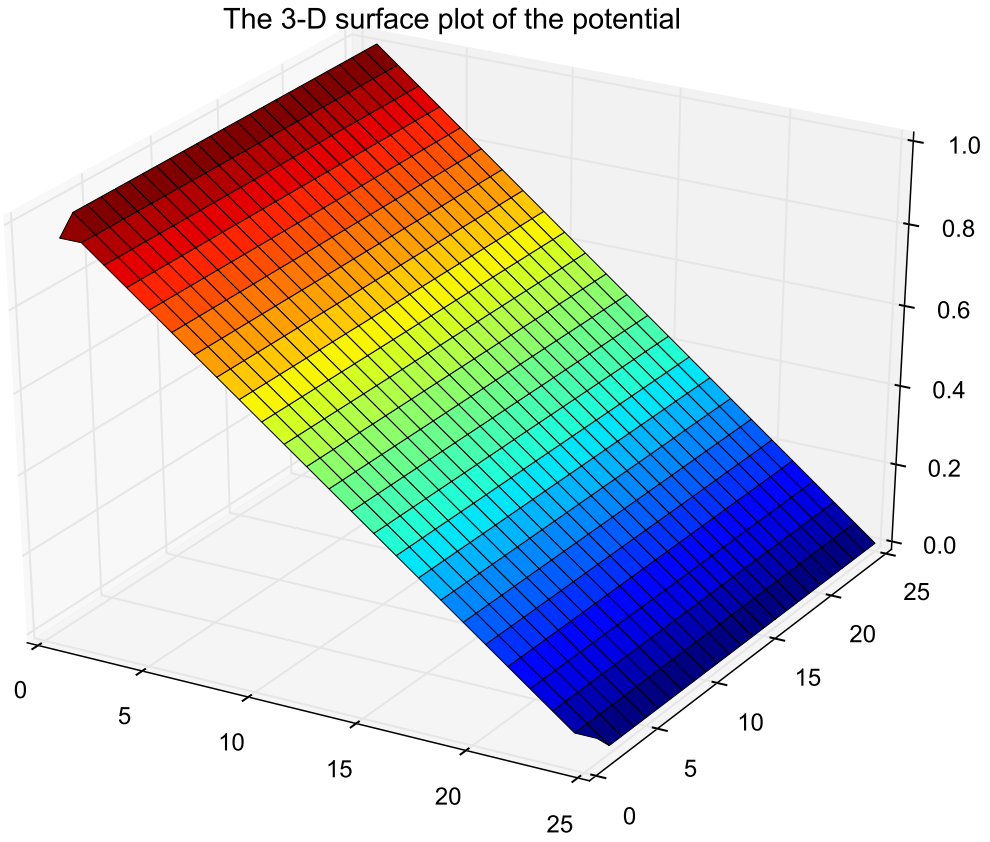
3.0.2 Error Curve Fitting for Electrode Length = Edge Length(i.e. 25)



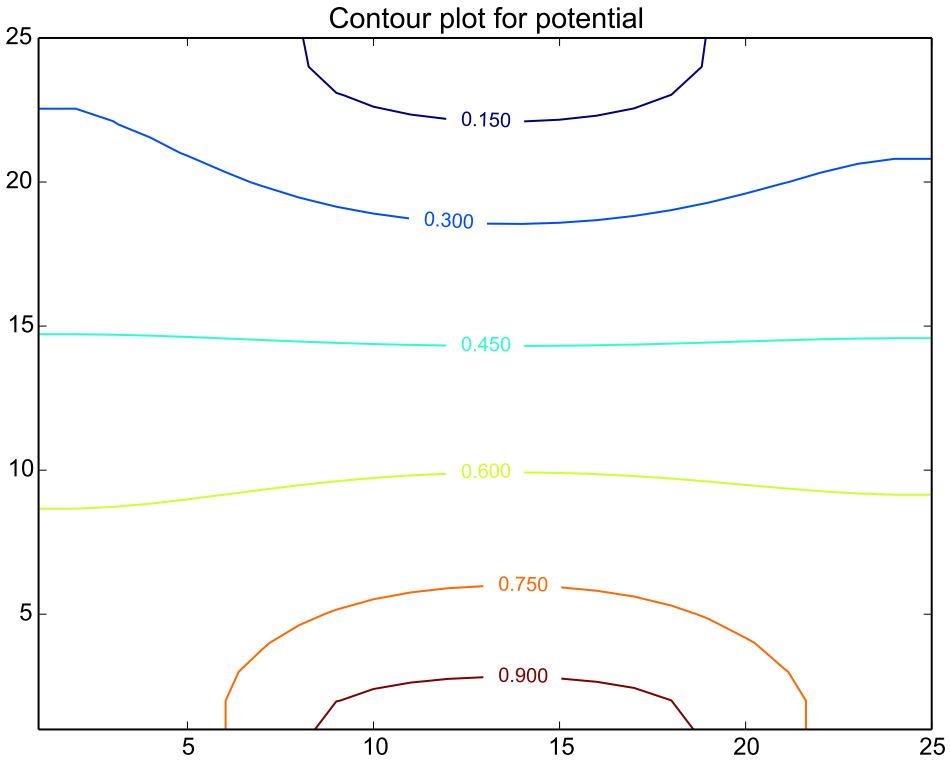
3.0.3 Surface Plot of Potential for Electrode Length = 10



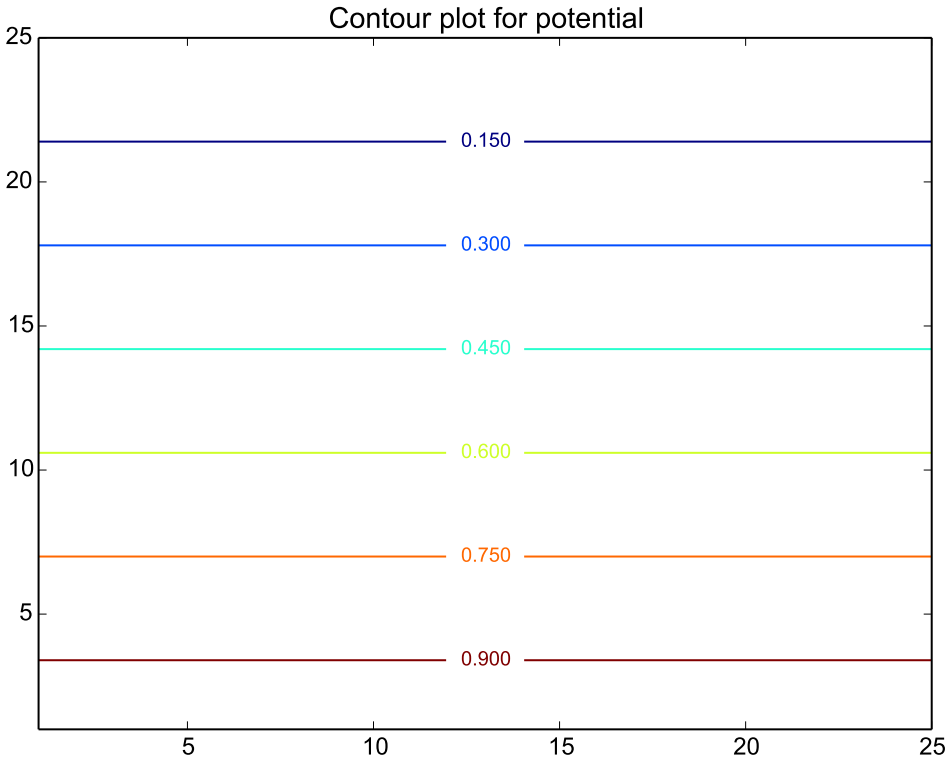
3.0.4 Surface Plot of Potential for Electrode Length = Edge Length(i.e. 25)



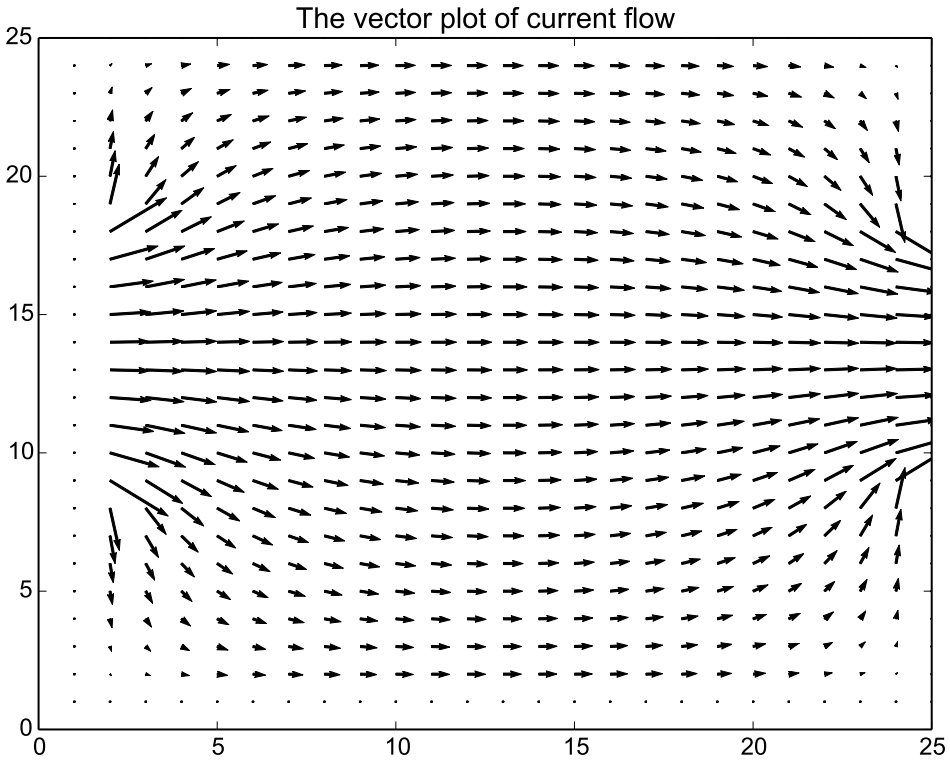
3.0.5 Contour Plot of Potential for Electrode Length = 10



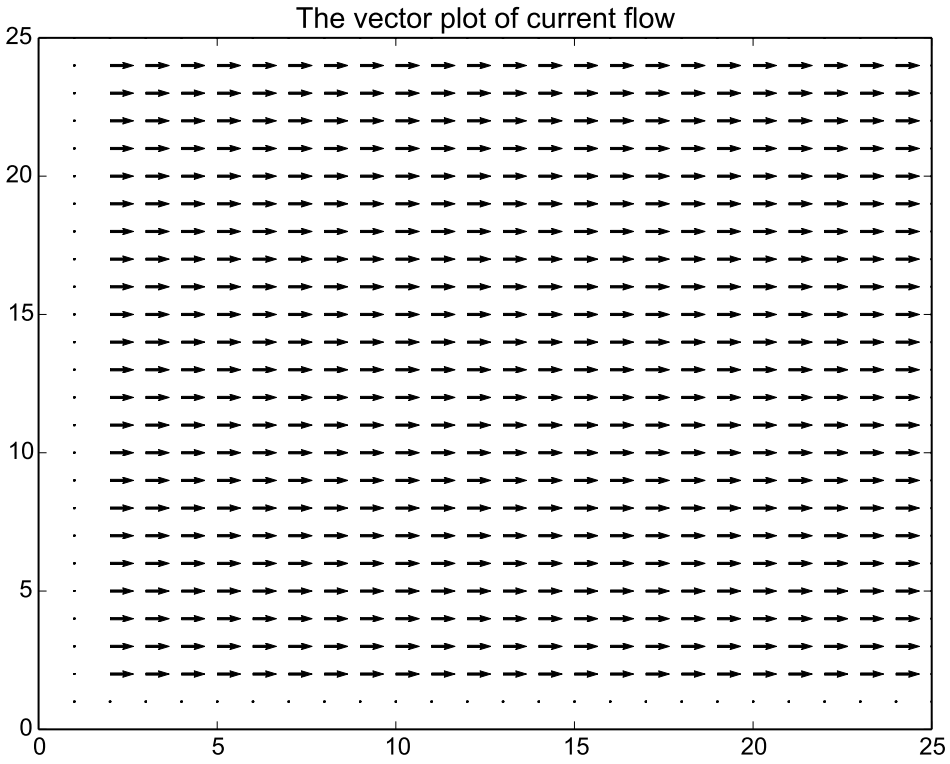
3.0.6 Contour Plot of Potential for Electrode Length = Edge Length(i.e. 25)



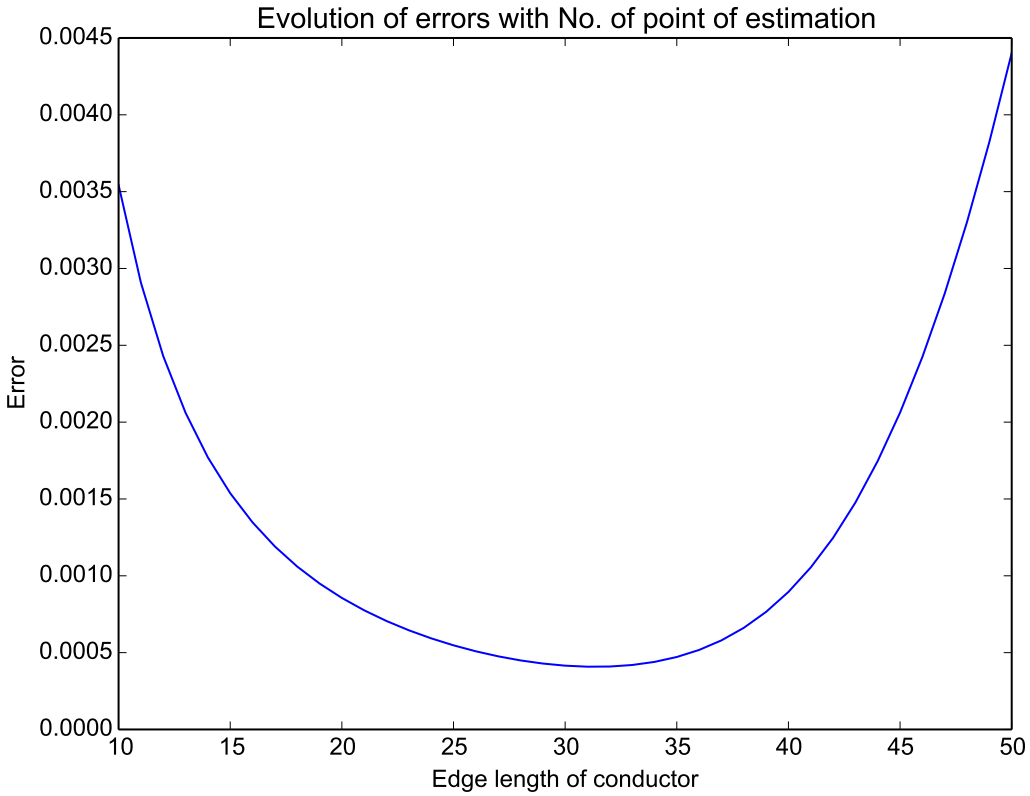
3.0.7 Current Density Plot of Potential for Electrode Length = 10



3.0.8 Current Density Plot of Potential for Electrode Length = Edge Length(i.e. 25)



3.0.9 Plot of Mean square error in potential Vs. Electrode Length(=EdgeLength)



3.1 ANALYSIS:

1. The resistance, when electrode length =10 is approximated to be: 1.439799 ohms.
2. On curve fitting for Least square approximation of error: A=0.002719, B=-0.004287 .
3. Final estimated error in potential(as in Section 1.3.1): 0.000120 .
4. For Niter: 2000 Iavj=0.958333 Idiff=0.000715 .
5. The resistance, when electrode length = edge length(i.e. 25) approximated to be: 1.043478 ohms.
6. Mean square error for Nx=25,Ny=25 and no. of iterations = 2000 is 0.000548.
7. Mean square error for Nx=50,Ny=50 and no. of iterations = 2000 is 0.004402.
8. Mean square error for Nx=100,Ny=100 and no. of iterations = 2000 is 0.076496.

4 RESULTS AND DISCUSSION:

4.1 .

This method is one of the slowest methods of reducing error in potential estimation because we are previously estimated potential(i.e. previous iteration) to estimate potential rather than a charcteristic of final potential to estimate potential.

4.2 .

In plot , the error is high for lower values of electrode length because we don't have known value of potential for enough points in the conductor.
The error is high for higher values of electrode length because the number of iterations, Niter=2000 is not high enough to compute potential accurately at that high value of electrode length.
Thus, the error vs. electrode length approaches a minima where the value of potential is known for enough points and the number of iterations is sufficient for that value of electrode length.