Assignment-4

M.Sai kumar-EE20B082

18-02-2022

1 Plotting counter plot.

Plot a contour plot of the potential in Figure 1. Mark the V = 1 region by marking those nodes red.I took radius as 8 units. A cylindrical 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 25 cm by 25 cm in size.

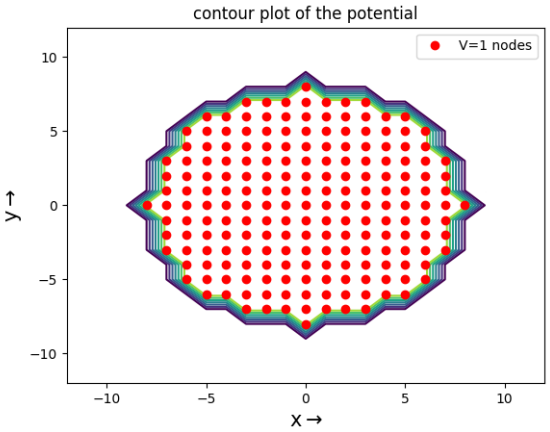

Figure 1: contour plot of the potential

2 Plot of error vs iterations.

2.1 Updating the potential

The potential at any point should be the average of its neighbours.The solution process is obvious. Guess anything you like for the solution. At each point, replace the potential by the average of its neighbours. Keep iterating till the solution converges (i.e., the maximum change in elements of *phi* is less than some tolerance

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

```
updating phi matrix
phi[1:-1,1:-1]=0.25*(phiold[1:-1,0:-2]+ phiold[1:-1,2:]+ phiold[0:-2,1:-1] +
```

At boundaries where the electrode is present, just put the value of electrode potential itself. At boundaries where there is no electrode, the gradient of *phi* should be tangential. This is implemented by requiring that *phi* should not vary in the normal direction.

2.2 Enforcing Boundary Conditions

The bottom boundary is grounded. The other 3 boundaries have a normal potential difference zero.

```
def boundary(phi,mask = np.where(X**2+Y**2<(0.35)**2)):
    #Left
    phi[:,0]=phi[:,1]
    #Right
    phi[:,Nx-1]=phi[:,Nx-2]
    #Top
    phi[0,:]=phi[1,:]
    #Bottom
    phi[Ny-1,:]=0
    #wire
    phi[mask]=1.0
    return phi
```

2.3 Calculating error after each iteration

```
err = np.zeros(Niter)
for k in range(Niter):
    phiold = phi.copy()
    phi = update_phi(phi,phiold)
    phi = boundary(phi)
    err[k] = np.max(np.abs(phi-phiold))
```

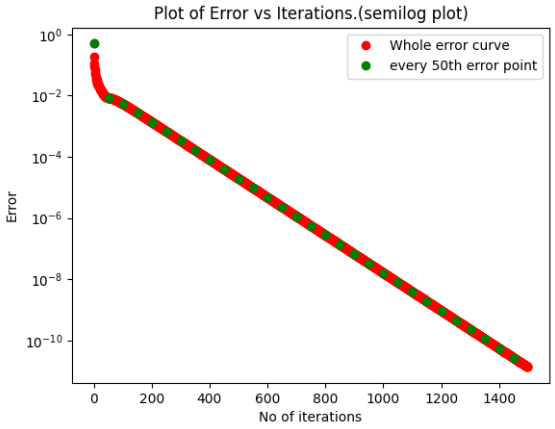

Figure 2: semilog plot of errors.

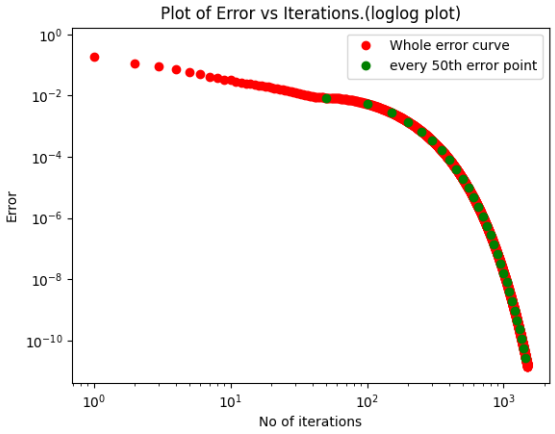

Figure 3: loglog plot of errors.

3 Plot of error and fit1, fit2

The error is a decaying potential for higher iterations.So we try to get a approximated function for given error.

$$y = Ae^{Bx} \tag{2}$$

We can estimate best log(A) and B with the least squares method.

$$log(y) = log(A) + Bx \tag{3}$$

Then we need to plot fitl for 0 to 500 iterations, Then fit2 from 500 iterations to end. And we need to plot error plot along with this plots. We can see that there is very less difference between the plots.

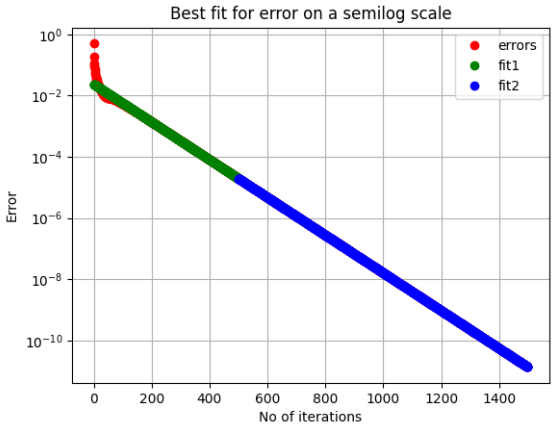

Figure 4: Best fit of error on semilog

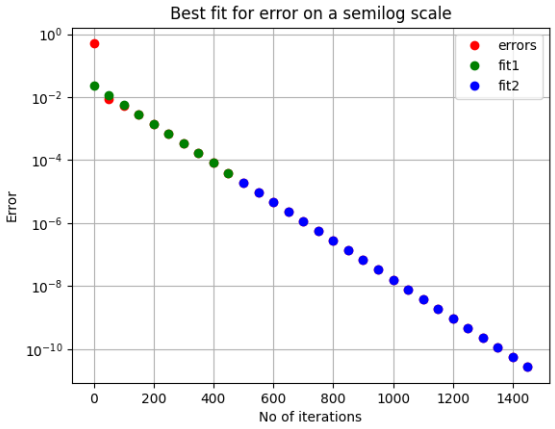

Figure 5: Best fit of error on semilog(every 50th iteration)

4 Plotting maximum error

The profile was changing very little every iteration, but it was continuously changing. So the cumulative error was still large. This method of solving Laplace's Equation is known to be one of the worst available. This is because of the very slow coefficient with which the error reduces.

$$= -A * exp(B(N + 0.5))/B \tag{4}$$

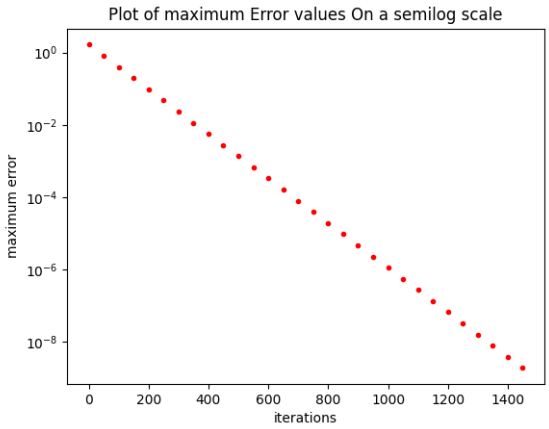

Figure 6: Cumulative error values on a log log scale

5 3D plot of potential

```
fig1=figure(4) # open a new figure
ax=p3.Axes3D(fig4) # Axes3D is the means to do a surface plot
title('The 3-D surface plot of the potential')
surf = ax.plot_surface(Y, X, phi.T, rstride=1, cstride=1, cmap=cm.jet,l
```

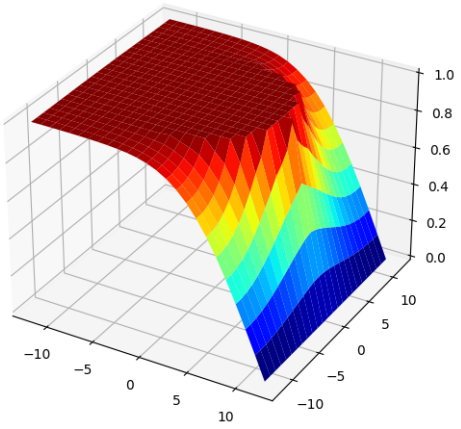

Figure 7: 3D plot of potential.

6 Plotting counter plot.

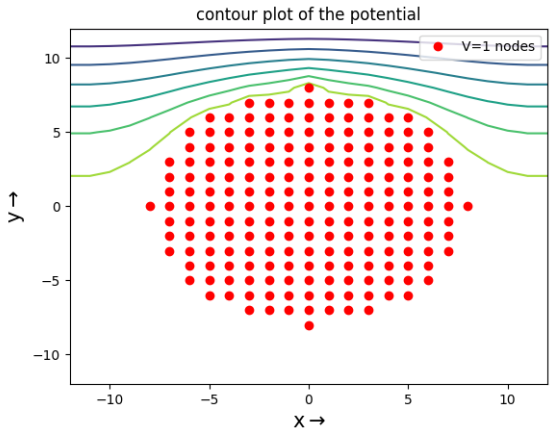

Figure 8: contour plot of the potential

7 Calculatoin and direction of current

obtain the currents requires computing the gradient. The actual value of *sigma* does not matter to the shape of the current profile, so we set it to unity. Then we get following equations if we assume some approximations for gradient.

$$J_{x,ij} = \frac{\phi_{i,j-1} - \phi i, j+1}{2} \tag{5}$$

$$J_{y,ij} = \frac{\phi_{i-1,j} - \phi i+1, j}{2} \tag{6}$$

Create the arrays Jx, Jy. Then call the quiver command.

```
quiver(y,x,Jy[::-1,:],Jx[::-1,:])
```

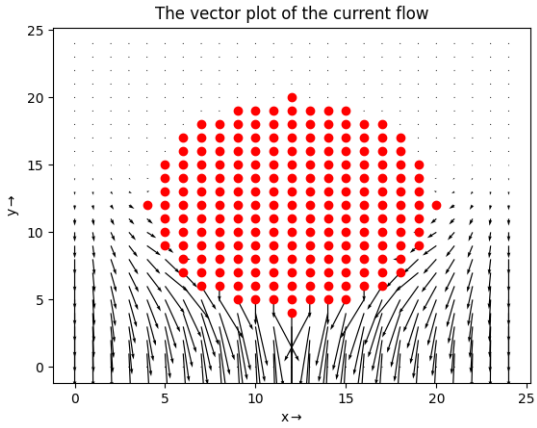

Figure 9: The vector plot of the current flow