EE2703 : Applied Programming Lab
Endsem

Ch Venkat Sai
EE18B042

February 14, 2021

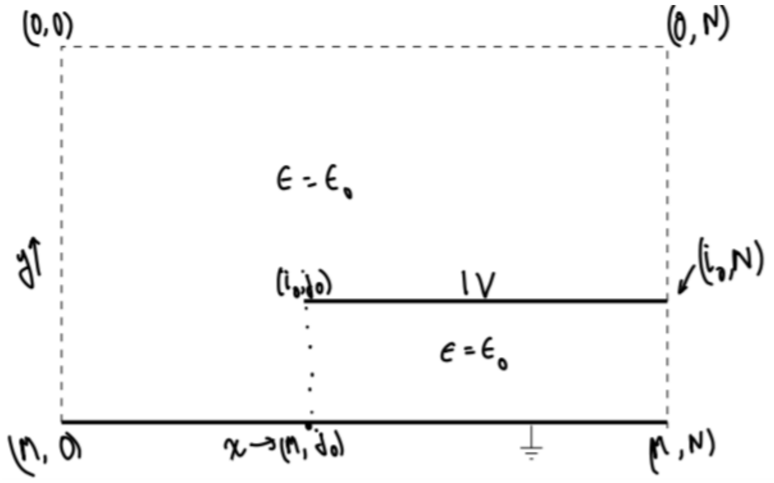


Figure 1: The setup

Overview

Time spent:6.5 hours The experiment is about:
We would like to model the fields of a capacitor surrounded by boundary where normal Electric field is zero.
Plotting the error in each iteration and finding best possible number of iterations for a limited error
Plotting the contour plot of the potential along with the vector plot of the fields.

Theory

A capacitor plate is placed at the bottom and connected to ground. Another capacitor plate which is smaller in size is placed at the top and is connected to a higher potential(in this case 1V). This creates an electric field. This setup surrounded by boundaries where normal electric field is zero. The setup is shown in the figure 1.

Now the electric field is the gradient of the potential,

$$\vec{E} = - \bigtriangledown \phi \tag{1}$$

and continuity of charge yields

$$\bigtriangledown \cdot \vec{j} = -\frac{\partial \rho}{\partial t} \tag{2}$$

Combining these equations and assuming a material with constant conductivity and DC currents flowing, we obtain

$$\bigtriangledown^2\phi = 0 \tag{3}$$

In 2-D the equation becomes:

$$\frac{\partial^2\phi}{\partial^2 x} + \frac{\partial^2\phi}{\partial^2 y} = 0 \tag{4}$$

Assuming ϕ is available at (x_i, y_j) we obtain,

$$\phi_i j = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi i_{,j-1}}{4} \tag{5}$$

At boundaries since the condition is to have no normal fields, the normal potentials must be the same. So for the boundary points we equal the potential it the points normal to them.

Define the Parameters

We have to define the position of the plate, dimensions of the boundaries and the number of iterations to be performed.

```
M = 30
N = 25                              #dimensions of boundary
i = 12
j = 10                              #capacitor location
Nit = 5000                          #given no. of iterations
```

Simulating array and Updating the Potential

First we create the ϕ array and initialize it to zero. And then we find the points which correspond to the upper and lower plate of capacitor and initiate the them to 1 and 0 respectfully. We find the points using where() function. We use python vectors to update the potential in each iteration using the equation we obtained finally and the boundary conditions. Using python vectors instead of for loops drastically reduces the time of execution.This part has been defined as a function for further uses with a change in i and no. of iterations.

```
x = linspace(int(0), int(N), N+1)
y = linspace(int(0), int(M), M+1)
X, Y = meshgrid(x, y)                  #creating a meshgrid

def getphi(i, Niter):
    phi = zeros([M+1, N+1])            #creating simulation array
    tt = where((Y == i) & (X >= j))    #points of top capacitor
    phi[tt] = 1                        #voltage = 1V
    mm = where(Y == M)                 #points of lower capacitor
    phi[mm] = 0                        #voltage = 0V
    errors = zeros(Niter)              #initialising errors
    x_scale = []
    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[0:-2,1:-1]+phi[2:
        phi[1:-1,0] = phi[1:-1,1]      # boundary conditions(first column)
        phi[1:-1,-1] = phi[1:-1,-2]    #last column
        phi[0,1:-1] = phi[1,1:-1]      #top row
        phi[mm] = 0                    #reassigning potentials
        phi[tt] = 1
        errors[k] = (abs(phi-oldphi)).max()    #error calculation
    return phi, errors
```

We have assigned 1 and 0 to the respective capacitor plates. Here in the vectorized code $phi[a,b]$, a corresponds to the rows and b to columns. The boundary conditions being potential at the top, left, right boundary are equal to the next row or column. And the bottom one stays at zero. While doing this the the electrode may get disturbed so we once again assign it to one. We also note the errors in each iteration,

Graph the results

We plot the semilogy plot of the errors using every data point. We see that the semilog plot is almost linear(Figure.1). So we assume the error varies as exponential and find the best fit for it using all the data. And by using this, we get the values of A and B.

```
mas, error = getphi(i, Nit)            #obtaining phi and error
x_scale = linspace(int(1), int(Nit), Nit)
semilogy(x_scale, error, '-r', label='errors semilogy')    #semilog plot for err
legend(loc='upper right')
```

```
xlabel(r'$N$',size=10)
ylabel(r'$errors$',size=10)
show()
s = []
errors1 = []
for k in range(Nit):
    s.append([1, x_scale[k]])
for k in range(Nit):
    errors1.append(log(error[k]))
errors1 = array(errors1)
s = array(s)
m = array(lstsq(s, errors1, rcond=None)[0])         # finding best fit1
A = exp(m[0])
B = m[1]
print(A,B)
```

Now given that the we have to obtain an accuracy to 2 digits. So we take the error to be 0.01. And we have the following formula to calculate the number of iterations.

$$Error = -\frac{A}{B}exp(B(N+0.5)) \tag{6}$$

After this gives us the number of iterations, we can calculate the new error and phi for the new iterations using the function used above.

```
Nnew = floor(np.log(-0.01*B/A)/B - 0.5)             #iteration no. for error<0.0
print(f"The new iteration number is {Nnew}")

mas, error = getphi(i, int(Nnew))       #new phi and error
```

Contour Plot of the Potential

Plotting the contour plot of the potential in order with the configuration in Figure1. Before plotting we interchange the Y part in the meshgrid and create a new one to fit our configuration.

```
x1 = linspace(int(0), int(N), N+1)      #creating a mesh grid of inverse dimensi
y1 = linspace(int(M), int(0), M+1)
X1,Y1 = meshgrid(x1,y1)
```

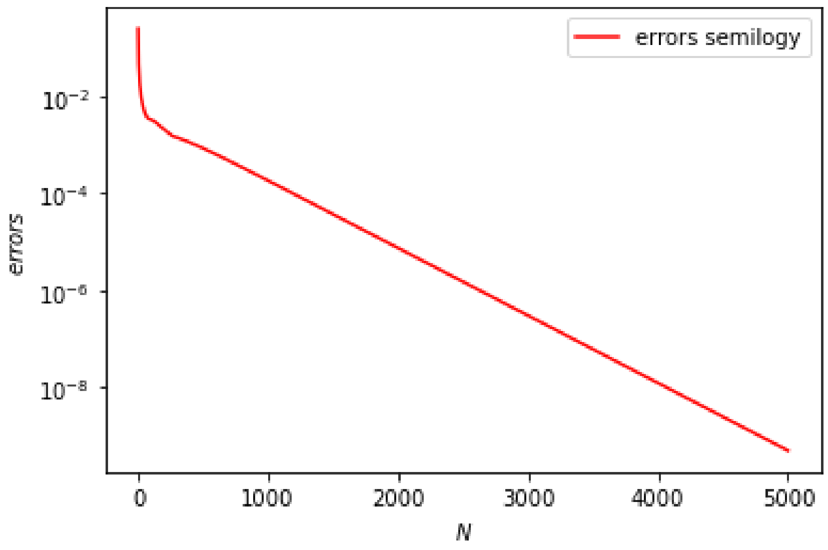


Figure 2: error semilogy, errors vs N

```
c = contourf(X1, Y1, mas)      #contour plot
colorbar(c)
xlabel(r'$x$',size=10)
ylabel(r'$30-y$',size=10)
show()
```

We get the following figure.

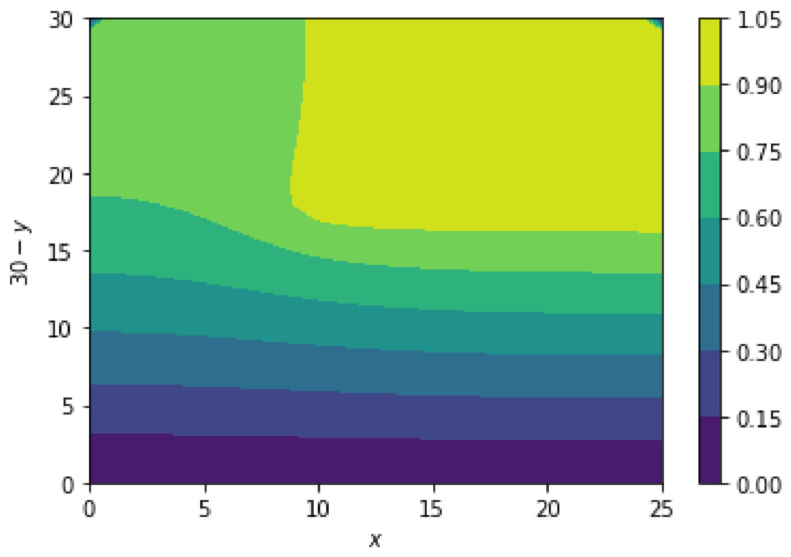


Figure 3: Contour plot

Vector Plot of fields

Now we plot the vector fields. This requires computing the gradient. So we calculate as follows. Our equations are:

$$e_x = -\frac{\partial \phi}{\partial x} \tag{7}$$

$$e_y = -\frac{\partial \phi}{\partial y} \tag{8}$$

This numerically translates to:

$$e_{x,ij} = \frac{\phi_{i,j-1} - \phi_{i,j+1}}{2} \tag{9}$$

$$e_{y,ij} = \frac{\phi_{i-1,j} - \phi_{i+1,j}}{2} \tag{10}$$

And then we use quiver to plot the current vector plot.

```
Jx = zeros([M+1, N+1])
Jy = zeros([M+1, N+1])
Jx[1:-1,1:-1] = 0.5*(mas[1:-1, 0:-2] - mas[1:-1, 2:])   #finding x-field vector
Jy[1:-1,1:-1] = 0.5*(mas[0:-2, 1:-1] - mas[2:, 1:-1])   #finding y-field vector
quiver(X1, Y1, Jx, -Jy)     #plotting field vectors
```

```
xlabel(r'$x$',size=10)
ylabel(r'$30-y$',size=10)
show()
```

The plot shows up like:

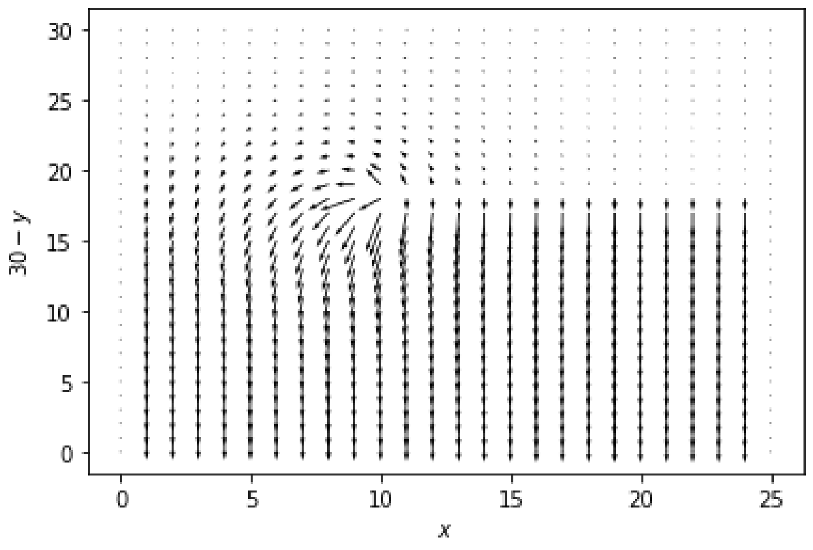


Figure 4: Vector field plot

Calculating charge and Dy

We now calculate the Dy vector just below the top plate and correspondingly calculate the charge which is accumulated below the top plate. We know the Ey field vector, we get the Dy vector by multiplying it with ϵ.

```
 e = 8.85e-12                 #value of epsilon
Dy = e*Jy[i+1,j:]            #calculating Dy
Q = 0                        #initiating charge
for k in range(len(Dy)):     #summing Dy
    Q += Dy[k]/len(Dy)       #averaging
print(f"The value of Dy is {Q}")
Q = Q*(N-j+1)                #multipying by area
print(f"The value of Q is {Q}")
print(f"The vector Dy is {Dy}")
```

After getting Dy vector we calculate the value of Dy by averaging it. And the to find the charge Q we multiply it with the area (due to Gauss law) as shown in code.

Calculting charge for different distances between the Capacitor

We now define a function to calcute the different charges. First with a change in distance we new ϕ and use it to calculate Jy and correspondingly Dy and charge Q.

```
def cap(l):                    #function to calculate Q
    las, sec = getphi(M-l, int(Nnew))       #new phi and error
    Jy[1:-1,1:-1] = 0.5*(las[0:-2, 1:-1] - las[2:, 1:-1])   #correspondin Jy or 
    e = 8.85e-12
    Dy = e*Jy[M-l+1,j:]               #corresponding Dy
    Q = 0
    for k in range(len(Dy)):
        Q += Dy[k]
    Q = Q/len(Dy)               #averaging
    Q = Q*(N-j+1)               #multipying by area
    return Q
```

And now we calculate the Q calculated from theory using the following formula.

$$Q_{theory} = \frac{\epsilon N - j_0}{i_0} V \tag{11}$$

```
def Qt(l):                     #function to calculate Qtheory
    Qt = e*(N-j)/(l)
    return Qt
```

And now we find the deviation in both the readings by taking the difference between the calculated and theoretical charge.

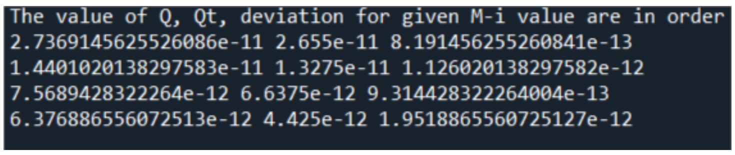
```
def dev(l):                    #function to calculate deviation
    d = cap(l) - Qt(l)
    return d
```

We print all these.

```
print(f"The value of Q, Qt, deviation for given M-i value are in order")
print(cap(5), Qt(5), dev(5))        #M-i=5
print(cap(10), Qt(10), dev(10))     #M-i=10
print(cap(20), Qt(20), dev(20))     #M-i=20
print(cap(30), Qt(30), dev(30))     #M-i=30
```

And we get the following table:

(M-i0)	(Qmeas (Coulumb))	(Qtheory (Coulumb))	(Qdev (Coulumb))
5	2.7310767420202653e-11	2.655e-11	7.607674202026517e-13
10	1.428977232247531e-11	1.3275e-11	1.0147723224753085e-12
20	7.537611764633292e-12	6.6375e-12	9.001117646332916e-13
30	6.027483475051889e-12	4.425e-12	1.6024834750518889e-12



```
The value of Q, Qt, deviation for given M-i value are in order
2.7369145625526086e-11 2.655e-11 8.191456255260841e-13
1.4401020138297583e-11 1.3275e-11 1.126020138297582e-12
7.5689428322264e-12 6.6375e-12 9.314428322264004e-13
6.376886556072513e-12 4.425e-12 1.9518865560725127e-12
```

Figure 5: Table

Explanation for Deviation

The deviation that occurs between the calculated between Q calculated and Q theoretical is due to the fringing effect. Because of the fringe effect the capacitance of a parallel plate capacitor is more than the capacitance calculated by the formula. Fringe effect occurs when the electric field extends the area of the overlap. As the capacitance is increased with the potential difference remainig the same, we have higher calculated charge. And thus the deviation.

Pseudo code

Import necessary libraries M - integer N - integer i - integer j - integer Nit - integer x = linearspace(0 to N) y = linearspace(0 to M) X , Y = meshgrid(x, y) define function getphi phi = zeros([M+1, N+1]) creating simulation array points1 = where((Y is equal to i (X is greater than j)) points of top capacitor phi[points1] = 1 voltage = 1V points2 = where(Y is equal to M) points of lower capacitor phi[points2] = 0 voltage = 0V errors = zeros(Niter) initialising errors

for k from 0 to Niger-1: oldphi = copy phi phi[all points] = 0.25*(phi(top+bottom + right+left)]) top row boundary cond phi[points1] = 0 reassigning potentials phi[points2] = 1 errors[k] = (abs(finalphi - origina)).max() error calculation return phi, errors mas, error = getphi(i, Nit) obtaining phi and error xscale = linearspace(int(1), int(Nit), Nit) plot semilogy

Calculate lstsq and A and B

Nnew = floor(log(-0.01*B/A)/B - 0.5) iteration no. for error¡0.01

mas, error = getphi(i, int(Nnew)) new phi and error

x1 = linearspace(int(0), int(N), N+1) creating a mesh grid of inverse dimensions y1 = linearspace(int(M), int(0), M+1) X1,Y1 = meshgrid(x1,y1)

plot contour

Jx = zeros([M+1, N+1]) Jy = zeros([M+1, N+1]) Jx[points] = 0.5*(mas[side] - mas[other side]) finding x-field vector Jy[points] = 0.5*(mas[side] - mas[other side]) finding y-field vector plot quiver

calculate Q and Dy

calculate Qtheory

deviation = Q - Qtheory