Indian Institute of Technology, Madras
Department of Electrical Engineering
Applied Programming Lab

Report

Final Exam

Nithin Uppalapati
EE18B035

01-08-2020

Contents

1. Abstract

Aim :
The aim of this assignment is to analyse the circuit which will be helpful in determining the level of fluid(dielectric) in a rectangular(cuboidal) tank of specified dimensions.

Given the dimensions of the tank, we should be able to measure the potential profile within the specified accuracy δ, and also we should measure Electric Field profile in the tank and should develop an algorithm to determine h from the observed resonant frequency.

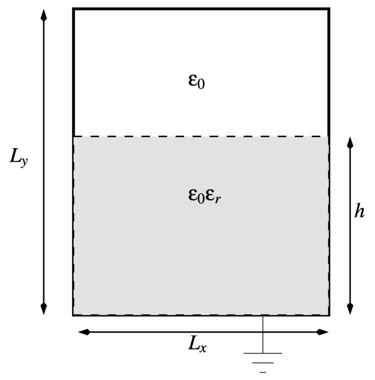
2. Introduction

Final Exam is based on:

- Solving the Laplace's equation by discretization of the differential equation in order to get the electric potential profile
- Analysing the error with every iteration, and fitting the error to a exponential function of the form: Ae^{Bx}, where x is the number of iteration performed
- Developing an algorithm for determining h from the resonant frequency of the series-RLC circuit
- Plotting graphs using the pylab library for both of the potential contours, errors in loglog scale and semilogy scale

1. Potential Contours
2. Plotting Electric Field vectors with the quiver command
3. Errors with each Iteration
4. Variation of $Q_{TopPlate}$ and Q_{Fluid} with the variation of level of Fluid in Tank.

Problem Description :
The sides and bottom of the tank are grounded while the top of the tank is connected to the sides via a series RLC circuit (the tank is the C) and an AC source.



Implementation :

- So, in order to find the electric potential in the tank, we solve the Laplace's equation under proper boundary conditions. [As done in the Assignment-5]
- And by calculating the gradient(difference) of the potential field, we calculate the Electric Field.
- As we have a dielectric boundary, we have to take care of that boundary condition too.
- And calculating the Charges on top plate of the tank and on the dielectric boundary by the application of Gauss' Law.
- Synthesizing an algorithm for determining h_{fluid} from resonant frequency
- Proving the continuity of D along the normal of the dielectric boundary

3. Theory

Electrodynamic Equations :
Let ϵ_o be the electric permittivity of free space, and ϵ_r be the dielectric constant of the dielectric fluid.
Then Electric field is the gradient of the potential,

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

and continuity of the Auxiliary field D,

$$\vec{D}.\hat{n_1} = \vec{D}.\hat{n_2}$$

That is the Normal component of D is same on both media. When expressed in terms of $\vec{E}$ the above equation translates to:

$$\epsilon_1\vec{E}.\hat{n_1} = \epsilon_2\vec{E}.\hat{n_2}$$

Where as, the parallel component of Electric is the same across the dielectric boundary.
As,

$$\vec{\nabla} \times \vec{E} = 0$$

So, we get,

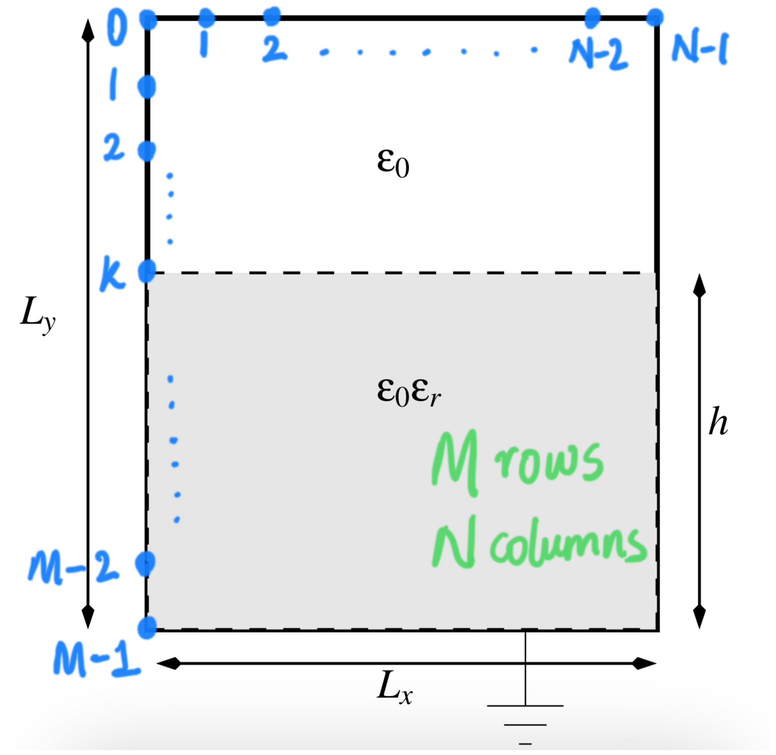
$$\vec{E_{1,//}} = \vec{E_{2,//}}$$

3.1 Symbols and Conventions of the Variables taken in the entire python code:

So, in order to solve this problem in python, we divide the tank into a mesh grid of **[M x N]** points, and also map the height to an index **k**.

- M, the number of nodes along x, including the boundary nodes
- N, the number of nodes along y, including the boundary nodes
- D, the distance between nodes (assumed same along x and along y)
- k, the height given as the index k corresponding to h
- d, The desired accuracy d

- Niter, the maximum number of iterations to complete.
- n_0, the actual number of iterations occurred
- q_top, The charge on top plate
- q_fluid, the charge on the dielectric boundary
- phi, Electric potential profile
- E_x, E_y ; Electric fields in x and y directions respectively
- errore, array of errors with each iteration



3.2 Calculating ϕ for the Region Inside Tank:

Laplace's equation is easily transformed into a difference equation.
The difference equation can be given as:

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

and this can be easily implemented in python without using nested for loop as shown below. This is possible to perform by using vectorized for loops.

Initializing the array of phi:

$$\phi = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{3.1}$$

```
phi=array([zeros(N)]*M)
phi[0]=1
```

where,

M are the number of rows of array, and N is the number of columns of array.

So, we input the desirable quantities into the *pot* function, and we also specify the accuracy needed for the potential profile.

So, once the accuracy is reached the function automatically quits, because of the else : break -line, and the number of iterations performed is stored in n_0.

If it didn't reached the desired accuracy, then the function iterates the calculation Niter times.

Below is the snippet of the function which calculates potential.

```
def pot(M,N,D,k,d,Niter):
phi=array([zeros(N)]*M)
phi[0]=1

for i in range(1,Niter+1):
if errore[i-1] >= (d) :
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:,1:-1])
phi[k,1:-1] = ( (e_r)*(phi[k+1,1:-1]) + phi[k-1,1:-1] )/(1+e_r)
#asserting boundary conditions
phi[-1]=0
phi[:,0]=0
phi[:,-1]=0
phi[0]=1
n_0=i+1
else: break
```

3.3 Errors in Each Iteration:

3.3.1 Calculation of Errors:

The coefficients are stored in the fashion,

$$errors = \begin{bmatrix} e_0 & e_1 & e_2 & e_3 & \cdots & e_{iterations} \end{bmatrix}$$

Where, e_i is the error while iterating the loop for i^{th} time.
And this calculated as follows:

```
errore=ones(Niter+1)
errore[i]=(abs(phi-oldphi)).max()  ##is executed in the for loop
```

So, in each iteration the function checks whether the error is greater than the specified accuracy, and also the function updates the values of errors in the array named *errore*.

3.3.2 Fitting the Entire Vector of Errors for iteration exceeding 300:

If we plot the variation of the errors with the number of iterations, we observe that, for over 300 iterations the errors vs iterations follow a straight line in semilogy plot. So, we want to fit the errors to the form:

$$e = Ae^{Bx}$$

So, in order to find the exponential fit to the errors, we use lstsq method for determining the coeficients. [As, did before in previous assignments]

```
err_coeff_300=lstsq( samples[300:] , log(err[300:]) ,rcond=-1)[0]
A_300=exp(err_coeff_300[0])
B_300=err_coeff_300[1]
```

So, as we now know the form of variation of errors with each iteration, we can extrapolate the value of error for very large number of iterations, tending to ∞.

3.4 Computing the Electric Field :

The Electric field $\vec{E}$, is defined as follows:

$$\vec{E_x} = (-\frac{\partial \phi}{\partial x})$$

And this is numerically translated as follows:

$$E_{x,ij} = \frac{1}{2D}(\phi_{i,j-1} - \phi_{i,j+1})$$
$$E_{y,ij} = \frac{1}{2D}(\phi_{i-1,j} - \phi_{i+1,j})$$

We are calculating the electric field because, it will easier for us to calculate the charges accumulated on the capacitor plates.

And this is implemented in python as follows:

```
E_x=array([zeros(N)]*M)
E_y=array([zeros(N)]*M)
E_x[1:-1,1:-1]=0.5*(1/D)*(phi[1:-1,0:-2]-phi[1:-1,2:])
E_y[1:-1,1:-1]=0.5*(1/D)*(phi[0:-2,1:-1]-phi[2:,1:-1])
E_y[-1]=(phi[-2])*(1/D)
E_x[:,0]=-1*phi[:,1]*(1/D)
E_x[:,-1]=phi[:,-2]*(1/D)
```

3.5 Calculation of Charges :

We can calculate the charges on the plates, by using Gauss' law.

$$\iint \vec{E}.\vec{dS} = \frac{q_{en}}{\epsilon}$$

And this can be simplified to as follows, where D = distance between two nodes in the mesh.

$$\sum E_{y,ij}.D\epsilon = q$$

So, we can sum up all the electric field vectors(instead of integration), and then multiply with the L_x in order to get the charge Q_{top}, which is charge per unit length.

```
 q_top=(sum(E_y[1]))*eps*D # +sum(E_x[0])
 q_fluid= ((sum(E_x[k:,0])-(sum(E_x[k:,-1]))-(sum(E_y[-1]))))*eps*D
```

3.5.1 Computation of Charges for various h :

As, we have to comput ethe charges, both Q_{Top} and Q_{Fluid}, for various h, we can remap the h to various indices vertically, which are different k's.

So, we can define a function which takes g as an argument, g=no. of divisions required and in our case g=10, as shown below:

```
def charges(g): # g is the no. of divisions required
points=linspace(M,0,g+1)
points=points[1:-1].round()
Q_TP  = zeros(g-1)
Q_FL  = zeros(g-1)
HEIGHT= zeros(g-1)
for i in range(0,g-1):
tmp=int(points[i])
z=pot(M,N,D,tmp,d,Niter)
Q_TP[i]  = z[5]
Q_FL[i]  = z[6]
HEIGHT[i]= Ly-tmp*D
return [Q_TP,Q_FL,HEIGHT]
```

The above function returns the arrays of Q_{Top}, Q_{Fluid} and h, and we can use them later for plotting Q_{Top} vs h and Q_{Fluid} vs h.

3.6 Algorithm to Determine h from the Resonant Frequency:

In order to determine h from the resonant frequency (given the values of L-inductance and ω - natural frequency), we have to first evaluate the value of the capacitance of the tank.

$$\omega = \frac{1}{\sqrt{LC}}$$

So, as we kn ow ω and L, we can compute C.
Assuming the voltage maintained by the tank to be of $1Volts$,
as,

$$Q = C.V$$

If V=1 Volts, as we know the value of capacitance we can also find the value of the amount of Q_{Top}.
So, the problem reduces to finding the value of h given the value of Q_{top}.
We can do this by the snippet shown below:

```
#w=charges(40)
#H_funcoeff = polyfit(w[0],w[2],5)
#H_func=poly1d(H_funcoeff)
#L=100  #inductance
#W=1 #resonant freq
#C=1/(L*W*W)
#H_est=H_func(C)  #capacitance
```

The above snippet is commented at the end of the code, and can uncomment and run it to test the estimated H.
What this does is, it create a 5th degree polynimal which fits the function,

$$H_{fluid} = poly(Q_{Top})$$

So, when we input a Q_{Top} based on the calculations of the resonant frequency, it gives us the estimated height, H_{est}.

3.7 Finding Ex and Ey at the centre of Tank:

For finding the field components at the centre of the tank, we can call the funciton and can find the corresponding values:

```
Ef_y1=E_y[k-1][int(N/2)]
Ef_y2=E_y[k+1][int(N/2)]*e_r
```

```
Ef_x1=E_x[k-1][int(N/2)]
Ef_x2=E_x[k+1][int(N/2)]
```

3.8 Angle of Electric Field at the Dielectric Boundary:

The change in angle of electric field can be obtained by, computing the incident angle (with the normal), and by refracted angle (with normal)

The angles are computed by, $arctan(E_x/E_y)$.

```
inci=arctan(E_x[k-1]/E_y[k-1])
refra=arctan(E_x[k+1]/E_y[k+1])

diff=inci-refra
```

And the relation between the angles is given by,
As stated earlier, in horizontal direction the parallel components of E are same in both dielectric media, and in vertical direction(along the normal),
$\epsilon_1.E_{1,y}$=$\epsilon_2.E_{2,y}$
We can prove that,

$$tan(i) = E_{1,x}/E_{1,y}$$
$$tan(r) = E_{2,x}/E_{2,y}$$
$$E_{1,x} = E_{2,x}$$
$$tan(i)E_{1,y} = tan(r)E_{2,y}$$
$$\Rightarrow \epsilon_2 tan(i) = \epsilon_1 tan(r)$$

And the above result can be proven by implementing it in python too... BUT, in centre of the tank, as i=0 and r=0, we get a RUMTIME ERROR.

```
ratio=tan(inci)/tan(refra)
```

And according to our equation we should get a value of 0.5 for the ratio in above snippet.
These are the runtime warnings as we encounter $tan(0)/tan(0)$

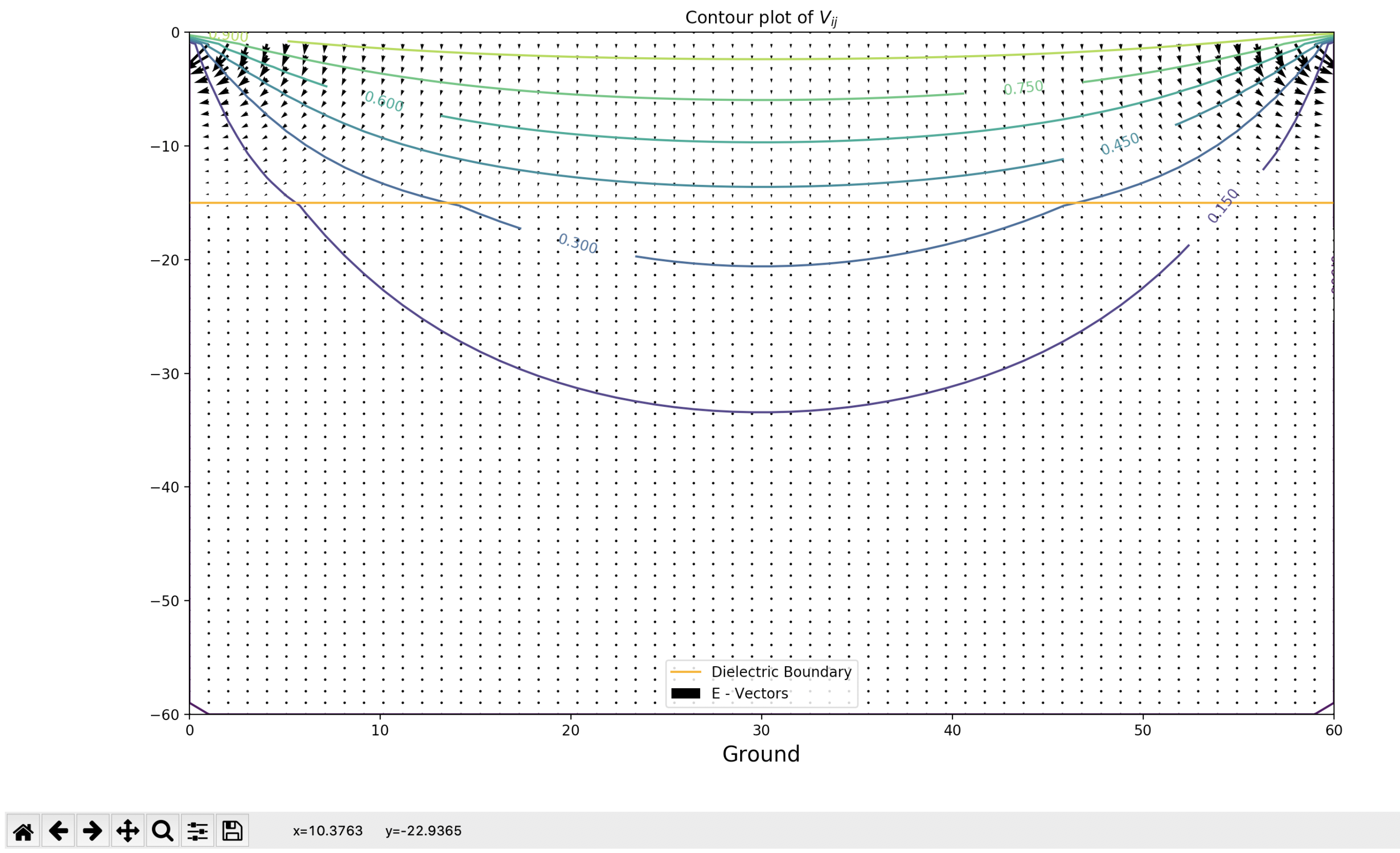
```
ee18b035.py:162: RuntimeWarning: divide by zero encountered in true_divide
  inci=arctan(E_x[k-1]/E_y[k-1])
ee18b035.py:163: RuntimeWarning: divide by zero encountered in true_divide
  refra=arctan(E_x[k+1]/E_y[k+1])
```

3.9 Plots of Functions and Coefficients:

```
n=arange(0,Niter,50)
```

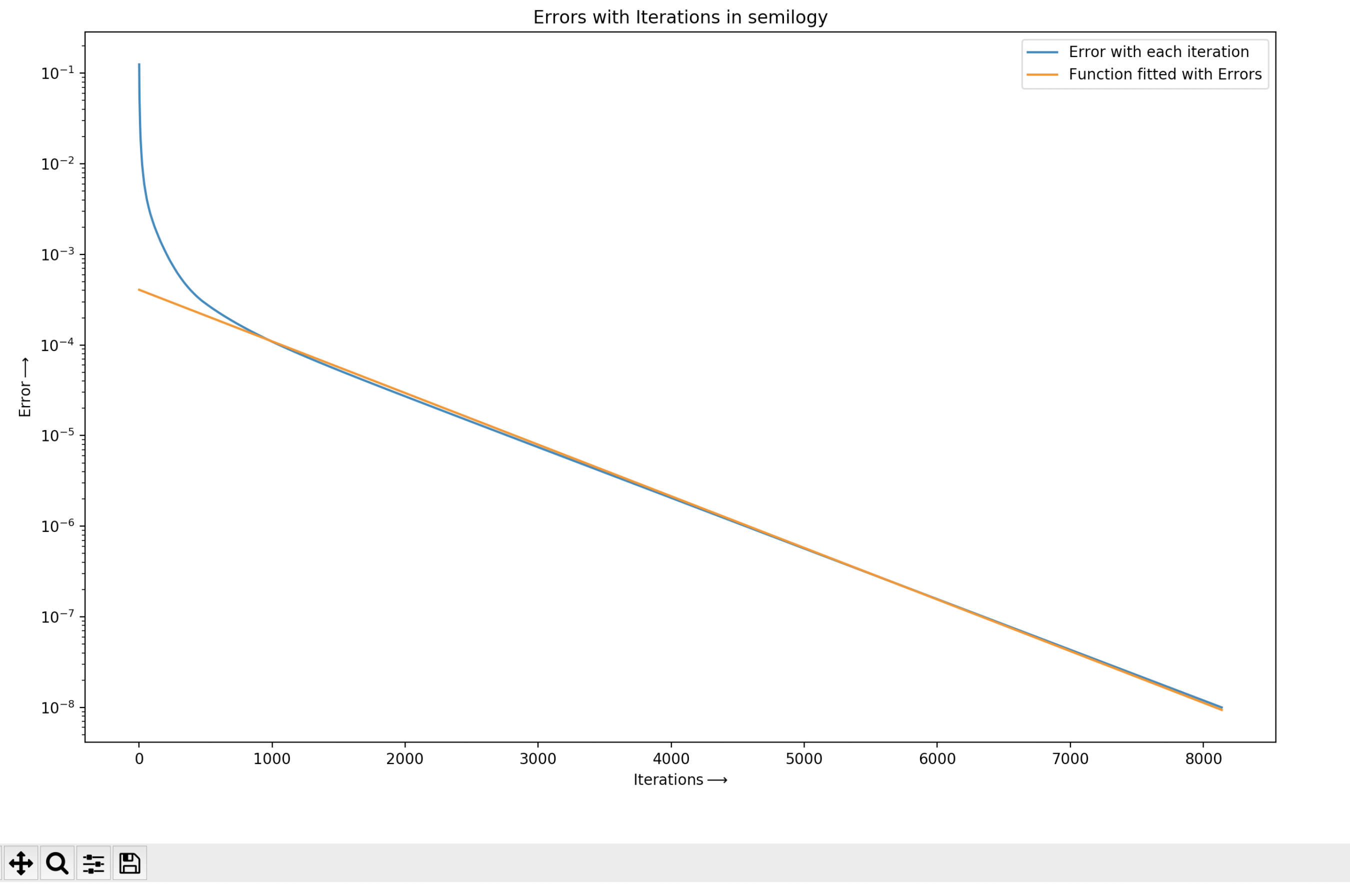
3.9.1 Plot of Electric Field and Potential :

```
ttitle('Vector plot of E-Field and Potential')
q=quiver(x,-y,E_x,-E_y, scale=1.5, scale_units='inches',label="E - Vectors")
cp = plt.contour(x,-y, phi, linestyles='-')
clabel(cp, inline=True,fontsize=10)
xlabel('Ground',size=15)
title(r'Contour plot of $V_{ij}$ ' )
plot(x,linspace(-1*k,-1*k,N),color='orange', label="Dielectric Boundary")
legend(loc="lower center")
```



3.9.2 Semilogy plot of Errors for all iterations along with the Estimated Functions:

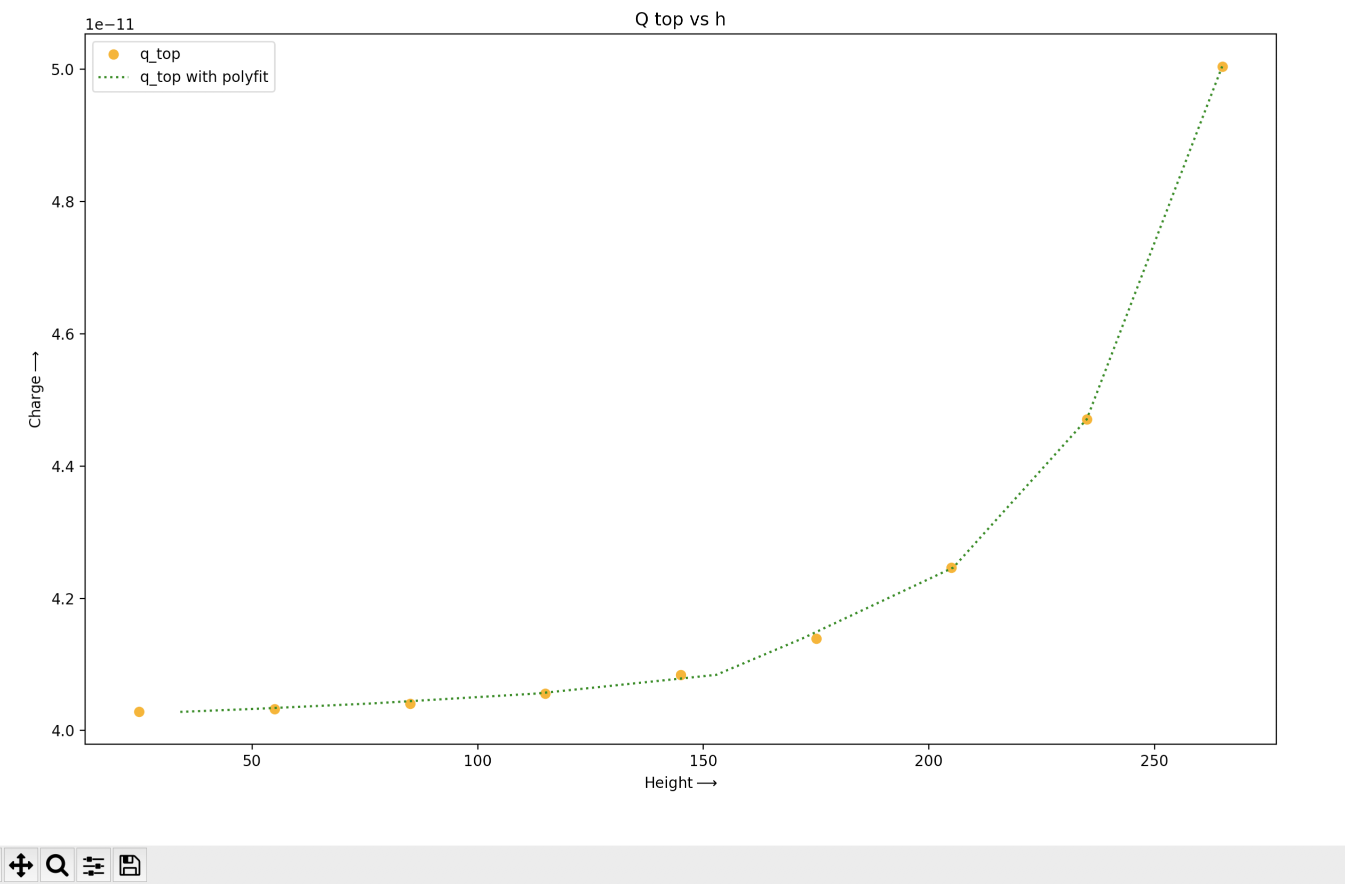
```
title('Errors with Iterations in semilogy')
n=arange(1,Nit-1,1)
semilogy(n,err,label="Error with each iteration")
semilogy(n,expo(n,A_300,B_300),label="Function fitted with Errors")
ylabel("Error"   '$\longrightarrow$')
xlabel("Iterations"   '$\longrightarrow$')
legend(loc="best")
```



3.9.3 Q_{top} vs h along with the polyfit function:

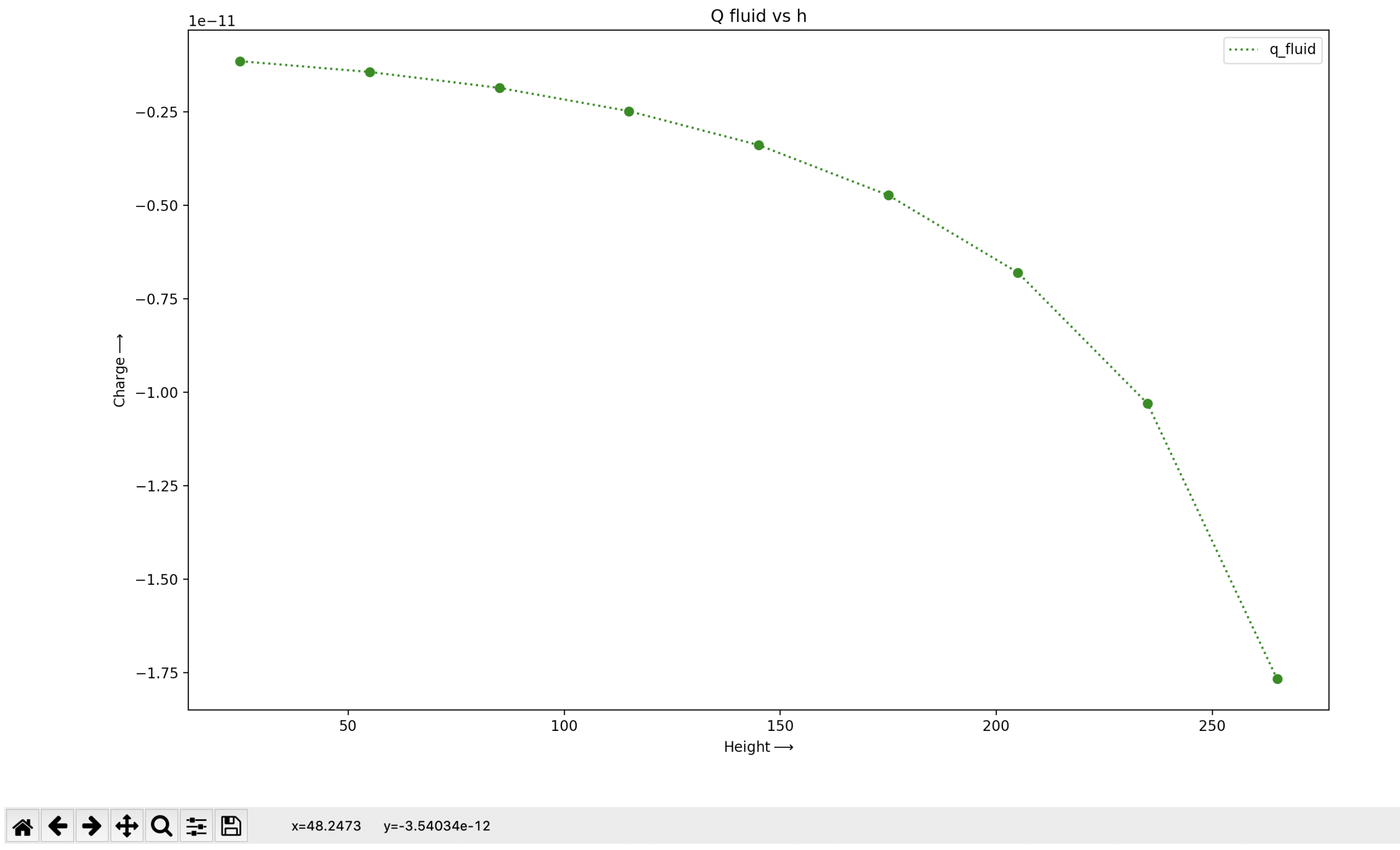
```
g=charges(10)
H_funcoeff = polyfit(g[0],g[2],5)
H_func=poly1d(H_funcoeff)
H_est=H_func(g[0])

title('Q top vs h')
plot(g[2],g[0],'o',color='orange', label="q_top")
plot(H_est,g[0],':',color='green',label="q_top with polyfit")
xlabel("Height"   '$\longrightarrow$')
ylabel("Charge"   '$\longrightarrow$')
legend(loc="best")
```



3.9.4 Q_{fluid} vs h :

```
title('Q fluid vs h')
plot(g[2],(g[1]),':',color='green', label="q_fluid")
plot(g[2],(g[1]),'o',color='green')
xlabel("Height"   '$\longrightarrow$')
ylabel("Charge"   '$\longrightarrow$')
legend(loc="best")
```



4. Conclusions

4.1 Does it agree with Snell's Law???

Actually, it should not agree with Snell's law, as Snell's law is valid for ElectroMagnetic Wave, whereas we are dealing with Pure Electric waves.

Instaed of satisfying Snell's law, it obeys other relation as derived above, which is,

$$\epsilon_2 tan(i) = \epsilon_1 tan(r)$$

And the output of the ratio of $tan(i)/tan(r)$array is shown as below:

```
[1.         0.56535706 0.56502048 0.56446766 0.5637104  0.5627645
 0.56164905 0.56038566 0.5589977  0.55750948 0.55594556 0.55433007
 0.55268626 0.55103602 0.54939962 0.54779555 0.54624039 0.54474881
 0.54333362 0.5420059  0.54077505 0.53964901 0.53863437 0.53773656
 0.53695995 0.53630804 0.53578354 0.53538848 0.53512434 0.53499204
 0.53499204 0.53512434 0.53538848 0.53578354 0.53630804 0.53695995
 0.53773656 0.53863437 0.53964901 0.54077505 0.5420059  0.54333362
 0.54474881 0.54624039 0.54779555 0.54939962 0.55103602 0.55268626
 0.55433007 0.55594556 0.55750948 0.5589977  0.56038566 0.56164905
 0.5627645  0.5637104  0.56446766 0.56502048 0.56535706
```

Which is consistent with the derived result above.

4.2 On Non-linearity of Charge variation with h:

As the tank arrangement is not as simple as that of the parallel plate capacitor, we can't expect a linear behaviour of the variation of q with h.

4.3 Parallelizing the Computation of ϕ:

As I used vector operations for the computation of the potential, it is more efficient than using nested for loops.

```
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])
phi[k,1:-1] = ( (e_r)*(phi[k+1,1:-1]) + phi[k-1,1:-1] )/(1+e_r)
```

4.4 The kink in the contour of the potential:

When we look at the potential contour plot we find a kink (non-differentiability), and this is because of change in the dielectric medium.