EE2703: Applied Programming Lab
End Semester Exam
Jan-May 2020

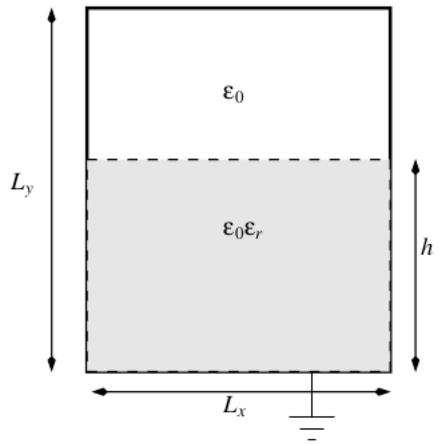
Arjun Menon Vadakkeveedu
EE18B104
Electrical Engineering
Indian Institute of Technology Madras

July 30, 2020

Contents

1 Introduction

In this program, a rectangular cross-section (closed, L_y = 20 cm, L_x = 10 cm) of a tank capacitor is analysed using Finite Difference Approximation of the Laplace Equation. The top wall of the tank is maintained at +1 V while the other three walls are maintained at 0 V. The walls are ideal conductors and the tank is filled up to a height h with a fluid of dielectric constant ϵ_r.



1.1 Approach

In order to find the effective capacitance of the tank, the potential inside the tank is first evaluated. For this purpose, Laplace's equation is discretised (with step size Δ into M and N points in y and x directions respectively) and solved iteratively.

$$\phi(m,n) = \frac{\phi(m+1,n) + \phi(m-1,n) + \phi(m,n+1) + \phi(m,n-1)}{4}$$
$$0 < m < M; 0 < n < N; m \neq k$$

$$\phi(k,n) = \frac{\epsilon_r \phi(k+1,n) + \phi(k-1,n)}{1+\epsilon_r}$$

The second relation is obtained as the normal component of $\vec{D}$ is continuous at the fluid-air interface.

Note: The convention followed throughout the code is that the +1 V plate is at y = 0 (m = 0) and y increases as m increases. Hence the 'bottom' plate maintained at 0V is at y = Ly and the fluid interface at y = Ly - h.

Once the potential is evaluated, the electric field is computed as the gradient of the obtained ϕ.

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

$$E_x = -\frac{\partial\phi}{\partial x}, E_y = -\frac{\partial\phi}{\partial y}$$

$$E_x \approx \frac{1}{\Delta}[\phi(x - \frac{\Delta}{2}, y) - \phi(x + \frac{\Delta}{2}, y)]$$

$$\phi(x - \frac{\Delta}{2}, y) \approx \frac{1}{2}[\phi(x - \frac{\Delta}{2}, y + \frac{\Delta}{2}) + \phi(x - \frac{\Delta}{2}, y - \frac{\Delta}{2})]$$

$$E_x \approx \frac{1}{2\Delta}[(\phi(x - \frac{\Delta}{2}, y + \frac{\Delta}{2}) + \phi(x - \frac{\Delta}{2}, y - \frac{\Delta}{2})) - (\phi(x + \frac{\Delta}{2}, y + \frac{\Delta}{2}) + \phi(x + \frac{\Delta}{2}, y - \frac{\Delta}{2})]$$

A similar four point gradient relation is obtained for E_y.
In order to evaluate the charge on the top plate (+1V), it is assumed that the electric field outside the tank is identically 0 (that is, there is negligible fringing outside the capacitor). Then, the normal boundary condition at the top plate yields:

$$\vec{D_{in}}.\hat{n} - 0 = \sigma_{top}$$

$\hat{n} = \hat{y}$. The total charge per unit length (in z-direction) on the top plate is then:

$$Q_{top} = \sum_x \epsilon_0 E_y(0, x)\Delta$$

The net capacitance of the tank (assuming the system is invariant along z-axis):

$$C = \frac{Q_{top} L_z}{V_{top}} = Q_{top}$$

where L_z is the length of the tank in the z-direction. Note that for the system to be invariant along the z-direction, L_z must be much greater than $\{L_x, L_y\}$ so as to avoid fringing in the z-direction. Q_{fluid} may be evaluated in a similar way, with ϵ_0 replaced by $\epsilon_0.\epsilon_r$ and the summation carried out over the appropriate regions.

1.2 Input and Output Formats

```
python3 EE18B104.py --help
usage: EE18B104.py [-h] [--varyH VARYH]

optional arguments:
  -h, --help     show this help message and exit
  --varyH VARYH  evaluate Q for h/Ly = {0.1, 0.2, ..., 0.9}

python3 EE18B104.py
Maximum bound for error = 2.111e-03
Last iteration value of error = 9.998e-07
Number of iterations computed = 11907
Ratio of Normal Components of D just above and beneath the fluid interface:
 Min. = 1.033e+00
 Max. = 1.033e+00
 Mean = 1.033e+00
```

---

2 Analysis

2.1 Solving for V- *Parts c, d*

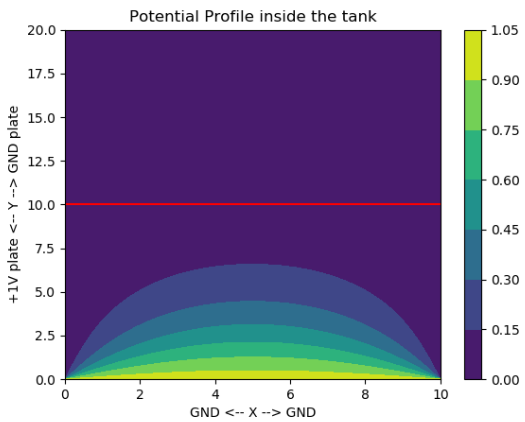
Code Snippet:

```
def solve_V(k, Nmax, delta = 1e-6):
 phi = np.zeros((M, N))
 phi[0, :] = 1
 errors = []
 for num_iter in range(Nmax):
  oldphi = phi.copy()
  phi[1:k, 1:-1] = 0.25*(phi[0:k-1, 1:-1] + phi[2:k+1, 1:-1] + \
    phi[1:k, 0:-2] + phi[1:k, 2:])
  phi[k+1:-1, 1:-1] = 0.25*(phi[k:-2, 1:-1] + phi[k+2:, 1:-1] + \
    phi[k+1:-1, 0:-2] + phi[k+1:-1, 2:])
  phi[k, 1:-1] = (eps_r*phi[k+1, 1:-1] + phi[k-1, 1:-1])/(1+eps_r)
  errors.append(np.max(abs(phi - oldphi)))
  if errors[-1] < delta:
   break
 return phi, errors
```

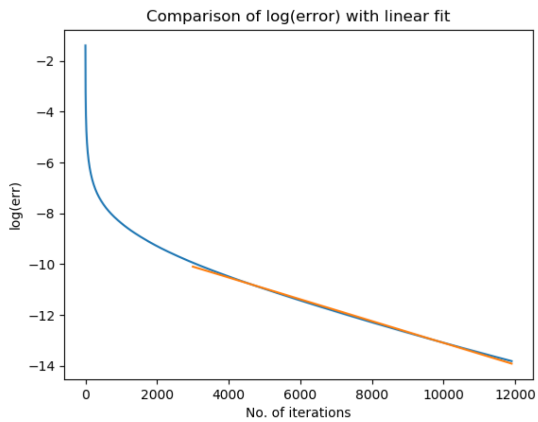
Python sub-arrays are used instead of Python for-loops as numpy-array operations, which are implemented directly using C for-loops, are faster.

The iterations are terminated when the error between consecutive ϕ's fall below a threshold, whose default value is 1e-6. This results in $\approx$ 12000 iterations.
The maximum bound for the cumulative error was estimated to be 2.111e-03. The approach followed in Assignment 5 was used for estimating the upper bound as log(error) is approximately linear w.r.t. N. The linear fit of log(error) was found using scipy.optimize.curve_fit() for N > 3000.

Plots:



Contour Plot of Potential; red hline marks the fluid interface



log(error) and linear fit versus no. of iterations

Notice that most of the potential is dropped for $0 < y < 7.5cm$. This is due to the side walls being at 0V. This is more apparent from the electric field profile.

2.2 Electric Field and Boundary Conditions- *Parts f, g*

Code Snippet:

```
def compute_grad(psi, step):
# Computes 4-point gradient of psi and returns -grad(psi)
 grad_x = np.zeros((M-1, N-1))
 grad_y = grad_x.copy()
```

```
grad_y[:, :] = ((psi[1:, 1:]+psi[1:, 0:-1]) - \
  (psi[0:-1, 1:]+psi[0:-1, 0:-1]))/(2*step)
grad_x[:, :] = ((psi[1:, 1:] + psi[0:-1, 1:])- \
  (psi[1:, 0:-1] + psi[0:-1, 0:-1]))/(2*step)
return grad_x*-1, grad_y*-1
```

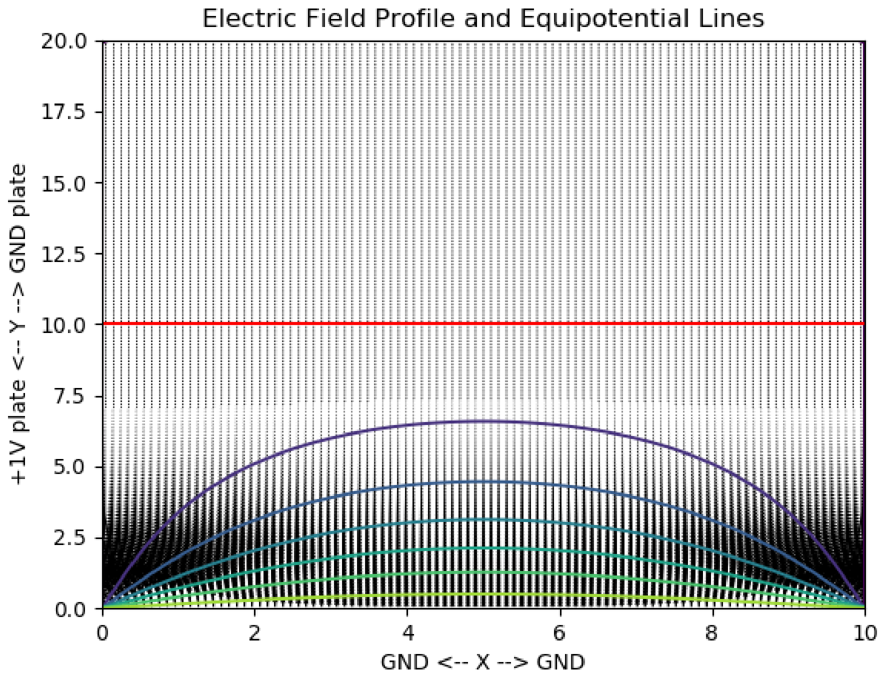


Figure 1: Electric Field Profile with Equipotential Lines; red hline marks the fluid interface

As expected, the electric field lines are crowded near the corners formed by the +1 V wall and the side walls. As V does not vary significantly near the middle of the tank, the Electric Field there is negligible.

Boundary Conditions: At the fluid-air interface inside the capacitor, the normal component of D is continuous. This is verified by taking the ratio of D[k+1, :] and D[k-1, :]:

```
# Code
D2_D1 = Ey[k-1,:]/(eps_r*Ey[k+1,:] + 1e-15)
print("Ratio of Normal Components of D just above and beneath \
  the fluid interface: \n Min. = {}\n Max. = {}\n \
```

```
  Mean = {}".format(roundf(np.min(D2_D1), precision= 3),\
  roundf(np.max(D2_D1), precision= 3), roundf(np.mean(D2_D1), precision= 3)))

# Output
Ratio of Normal Components of D just above and beneath the fluid interface:
 Min. = 1.033e+00
 Max. = 1.033e+00
 Mean = 1.033e+00
```

Further, at the walls of the tank, the normal component of D is discontinuous by an amount of σ, the surface charge density. As mentioned above, this can be used to estimate the charge stored on the walls.

Verification of Snell's Law: Snell's Law is satisfied by plane waves incident on an interface separating two media with different refractive indices, i.e.- dielectric constants:

$$\sqrt{\epsilon_1} sin(\theta_1) = \sqrt{\epsilon_2} sin(\theta_2)$$
$$\implies \frac{sin(\theta_1)}{sin(\theta_2)} = \sqrt{\frac{\epsilon_2}{\epsilon_1}}$$

Taking medium 1 as air and medium 2 as fluid, the above expression should hence be equal to $\sqrt{2}$. Since the electric field inside the capacitor is not a wave (magnetic field is static wrt time), Snell's law will not be satisfied. This is verified by evaluating the ratio of sines of the angles of incidence at y = k over x $\in (0, L_x)$.

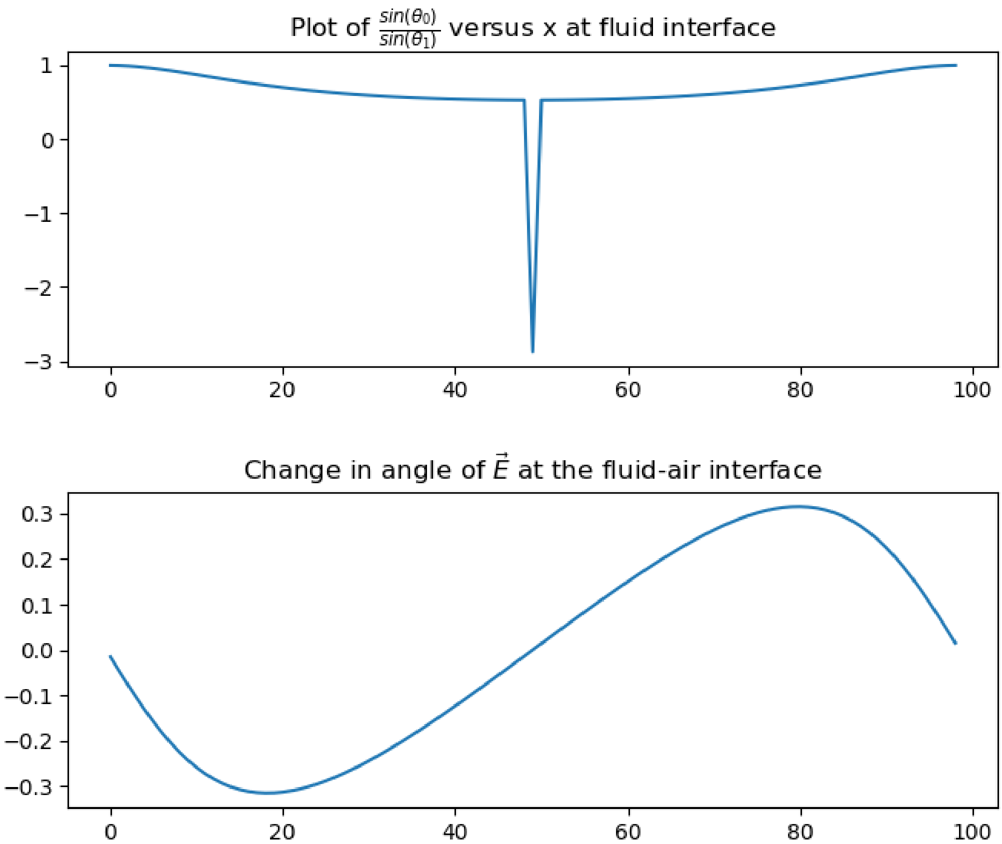


Figure 2: Behaviour of angle($\vec{E}$) at the interface

The ratio varies as x varies. The graph is symmetric about the midpoint (x = 5cm) due to the symmetric nature of the system and electric field profile.

2.3 Charge Stored- *Parts e, b*

Code Snippet:

```
Q_top = []
Q_fluid = []
H = Ly*np.linspace(0.1, 0.9, 9)
for ht in H:
  k, __, Ex, Ey, __ = eval_V_E(ht, Nmax = 5000)
  Q_top.append(np.sum(Ey[0, :]*step*eps0))
  Q_fluid.append(-1*eps0*eps_r*step*(np.sum((Ex[k+1:, -1] \
    - Ex[k+1:, 0]))+np.sum(Ey[-1, :])))
```

```
Cap = Q_top.copy()  # C = Q/V, V = 1 volt
# empirical C-h fit:
fit_data = 1/np.sqrt(Cap)
fit_coeffs = np.polyfit(H, fit_data, 5)
# Fitting 5th order polynomial to 1/sqrt(Q) (linear function of w_o) v/s h
fit_C = poly_5(H, fit_coeffs)
# poly_5 is a helper function to evaluate the polynomial value at H
```

The charge per unit depth (along z-axis) Q_{top} stored on the top plate of the tank (positive plate) does not vary significantly for $h \in (0.1L_y, 0.6L_y)$ and then rapidly increases for h $> 0.6L_y$.
As h increases beyond $0.6L_y$, a significant amount of the electric field begins to flow into the dielectric medium. As the capacitance of a dielectric medium is greater than that of vacuum, the net charge stored increases.
The rise in stored charge is non-linear for high h as the amount of field lines flowing into the dielectric medium increases in a non-linear fashion.

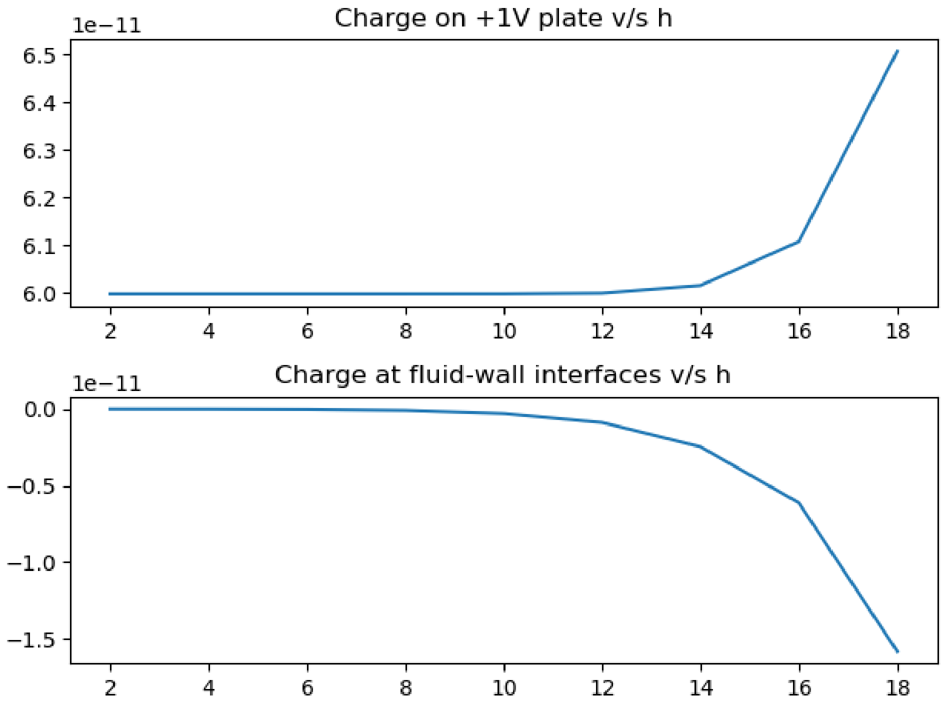


Figure 3: Variation of Q_{top} and Q_{fluid} with h

The charge stored on the +1V plate is positive and equal in magnitude to the charge stored on the other three walls. Q_{fluid} is negative and less than

Q_{top} in magnitude as a majority of the negative charge is stored on the side walls bounding the air medium.

2.3.1 Equivalent Capacitance and relation between ω_0 and h

The equivalent capacitance is equal to: $C = \frac{Q_{top} L_z}{V} = Q_{top} L_z$.

For a series LCR circuit with low damping factor (effectively, low R) has a resonant frequency at $\omega_0 \approx \frac{1}{\sqrt{LC}}$. Since finding a closed form analytic expression relating ω_0 and h is not computationally efficient, the relation between the two quantities is estimated through polynomial regression:

$$\omega_0 = f(h)$$
$$\omega_0 \approx a_0 + a_1.h + a_2.h^2 + ... + a_n.h^n$$
$$\omega_0 = \frac{\lambda}{\sqrt{C}}$$

A polynomial mapping h to ω_0 was considered instead of a map from ω_0 to h as the former gave better results. Upon finding the best fit, one may estimate h given ω_0 and L using an appropriate root finding method. Since $\frac{1}{\sqrt{C_z}}$, where C_z is the capacitance per unit depth ($= \frac{C}{L_z} = Q_{top}$) is a linear function of ω_0, the fitting (and resulting plots) are for $\frac{1}{\sqrt{C_z}}$ versus h.

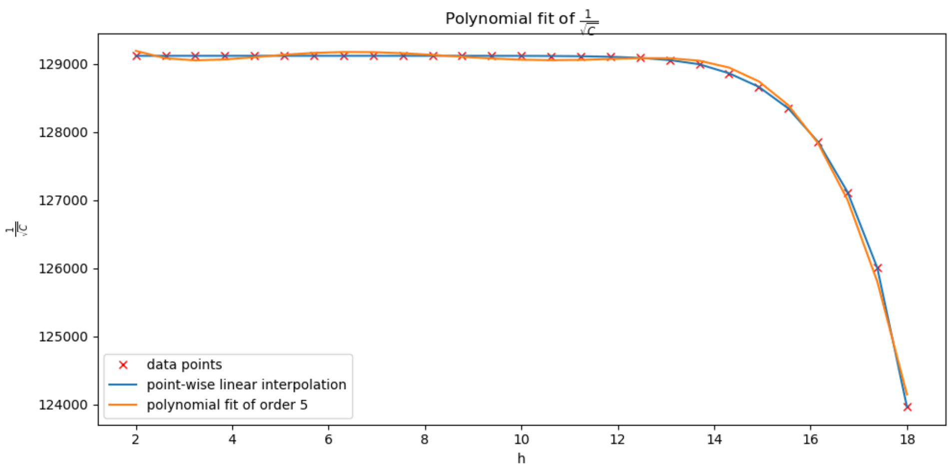


Figure 4: Variation of $\frac{1}{\sqrt{C}}$ with h

The polynomial fit was obtained using the numpy polyfit function.

```
'''
ALGORITHM TO FIND h GIVEN w_0, L AND L_z (depth along z-axis):

def poly_5(h, z):
# Returns the polynomial fit evaluated at h
return z[0]*(h**5) + z[1]*(h**4) + z[2]*(h**3) + z[3]*(h**2) + \
      z[4]*(h) + z[5]

Assumption: The system is invariant along z-axis, else, fringing will
take place at the edges of the walls // to x-y plane
w_0 = 1/sqrt(L*L_z*Cap) = poly_5(h, fit_coeffs)/sqrt(L*L_z)

Let w_0 = poly_5(h, w_coeffs); w_coeffs are coefficients of polynomial
map from h to w_0
=> w_coeffs = fit_coeffs/sqrt(L*L_z)

h = real root of poly_5(H, w_roots) - w_0 that belongs to the range (0, Ly)
Root may be found using np.roots function

Note: Due to small ripple in the polynomial fit of 1/sqrt(C) versus h in
the range (0, 0.6Ly), multiple roots may be obtained in the significant
range using this method. The actual mapping from w_0 to h must have atmost
one root in (0, Ly) as the behaviour is monotonic in this range.
'''
```

⋆ ⋆ ⋆