NUMERICAL SOLUTION TO 3D LAPLACE'S EQUATION OF ELECTRIC POTENTIAL

Álvaro Cauqui Diaz

Before we start, I chose to answer the questions on the go, without dedicating a specific section to answer the questions, wherever I think have written an answer for some question, I reference it with the question's number, e.g.:**(Question 1A)**

Introduction

Following Maxwell's equations, an Electric field must always satisfy the following:

$$\nabla \cdot E = \frac{\rho}{\epsilon} \tag{1}$$

Which, if one acknowledges that $E = -\Delta V$, can write as:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = \frac{\rho}{\epsilon} \tag{2}$$

The objective is to solve the latter PDE (called poisson's equation) numerically to simulate the electric potential around a finite-plate capacitor, with some boundary conditions that will be stated later on.By assuming that all the charges are contained inside the plates and using proper boundary conditions, we can simplify the equation to:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0$$

This last equation is known as Laplace's equation, and will in fact be the one being solved in the simulation.
The method of choice is the finite difference method.I will start by creating a three dimensional descrete 'mesh' of numbers in which each point is called a **node** and represents a point in space. Each node separated by a stepsize h.
In numerical analysis, one way to approximate derivatives is called the **finite difference method**, and uses the following approximation:

$$\left(\frac{\partial^2 V}{\partial x^2}\right) = \frac{V(x+\Delta x, y, z) - 2V(x, y, x) + V(x-\Delta x, y, z)}{\Delta x^2}$$

Which if we use the subindices i,j,k to designate a point in the mesh which is i steps in the x direction, j steps in the y direction and k steps in the z direction becomes:

$$\left(\frac{\partial^2 V}{\partial x^2}\right) = \frac{V_{i+1,j,k} - 2V_{i,j,k} + V_{i-1,j,k}}{\Delta h^2}$$

and analogously:

$$\left(\frac{\partial^2 V}{\partial y^2}\right) = \frac{V_{i,j+1,k} - 2V_{i,j,k} + V_{i,j-1,k}}{\Delta h^2}$$
$$\left(\frac{\partial^2 V}{\partial z^2}\right) = \frac{V_{i,j,k+1} - 2V_{i,j,k} + V_{i,j,k-1}}{\Delta h^2}$$

And the equation being solved becomes:

$$\frac{V_{i+1,j,k} - 2V_{i,j,k} + V_{i-1,j,k}}{\Delta h^2} + \frac{V_{i,j+1,k} - 2V_{i,j,k} + V_{i,j-1,k}}{\Delta h^2} + \frac{V_{i,j,k+1} - 2V_{i,j,k} + V_{i,j,k-1}}{\Delta h^2} = 0$$

Which can be rearranged as:

$$\frac{V_{i+1,j,k} + V_{i-1,j,k} + V_{i,j+1,k} + V_{i,j-1,k} + V_{i,j,k+1} + V_{i,j,k-1}}{6} = V_{i,j,k} \tag{3}$$

Which allows us to calculate V at a point i,j,k using V at all the surrounding points.In fact, the equation states that the potential at that point is the **mean** of all the values of the potential around that point.
Thus, in essence, the method is to start with some initial guess $V(x, y, z)$ and via the boundary conditions that we impose, itinerate equation (3) until the result converges to a V(x,y,z) for all the points in the mesh.

Aim

- Create a Python algorithm to solve Laplace's equation via the finite differences method
- use the numerical solution to produce plots of the potential around the capacitor along different directions and planes.
- To compare the numerical simulation to the theoretical potential around a **infinite** plate capacitor.
- Analysis of results.

Method description

1. Parametrization
2. Creation of the domain of the simulation.
3. Boundary conditions
4. Finite differences algorithm

1.Parametrization

We will chose the system to be a parallel plate capacitor inside a conducting box of size $L_x = 10cm$,$L_y = 15cm$,$L_z = 30cm$.The plates of the capacitor have sizes $L_y = 5cm$,$L_z = 10cm$ and are separated by a distance of $d = 1cm$. The stepsize h will be 0.1cm and the plates are at potentials V_1 and V_2.If we set the origin of coordinates between the plates, at the center of the conducting box:

```
import numpy as np
#%%
#**46,96
#parametrization
h= 0.1
xmin = -5

xmax = 5
ymin=-7.5
ymax = 7.5
zmin = -15
zmax = 15
V1= 10
V2 = -5
rtol=0.1
Nmax=500
```

2.Creation of the domain ('mesh') of the simulation & boundary conditions

For now to create our mesh we just need to define the nodes along the three spatial directions.I will do this by creating three 1D arrays of n points ranging from the lowest to the highest values along each direction separated by the stepsize h:

```
x = np.arange(xmin,xmax+h,h)
y = np.arange(ymin,ymax+h,h)
z = np.arange(zmin,zmax+h,h)
```

However, it is convenient to make now an array V containing the values of the potential at all points, where we will set our initial guess and set our boundary conditions.I made my initial guess to be 0, thus I start the simulation with an array which is zero everywhere except at the points where there are boundary conditions.I made the array V the same size as the mesh.

```
n=len(x)  #size of x-dimension
m=len(y)  #size of y-dimension
k=len(z)  #size of z-dimension
V=np.zeros((n,m,k)) #creation of a matrix of size n,m,k
V[45,50:101,100:201]=V1 #potential at location of the first plate in the mesh
V[55,50:101,100:201]=V2 #potential at the location of the second plate in the
    mesh
```

```
V[0,:,:]=0 #potential of the conducting box
V[100,:,:]=0 #potential of the conducting box
V[:,150,:]=0 #potential of the conducting box
V[:,0,:]=0 #potential of the conducting box
V[:,:,0]=0 #potential of the conducting box
V[:,:,300]=0 #potential of the conducting box
```

3.Finite differences implementation

Lastly, I want the algorithm to not visit the points where i have set boundary conditions, since they must remain constant throughout.I implemented this by creating a list containing only the points where there are no boundary conditions, so that the algorithm can access it and only calculate the values of V where they need to be calculated.
I did this by first creating a list containing the indices of all the nodes in the mesh:

```
allnodes=[]
for t in range(1,100):
    for y in range(1,150):
        for u in range(1,300):
            allnodes.append([t,y,u])
```

And then i created another list containing only the indices of the points with some boundary condition associated:

```
boundarynodesV1= []
for e in range(50,101):
    for f in range(100,201):
        boundarynodesV1.append([45,e,f])

boundarynodesV2=[]
for o in range(50,101):
    for p in range(100,201):
        boundarynodesV2.append([55,o,p])
```

Then, from the list containing all nodes, I selected only the points which where not on the list containing nodes with boundary conditions and **saved** that list:

```
calculatednodes1=[x for x in allnodes if x not in boundarynodesV1]
calculatednodes=[x for x in calculatednodes1 if x not in boundarynodesV2]
#%%
np.save("calculated nodes",calculatednodes)
```

Now, for the implementation of the algorithm itself, I made a algorithm that would visit each point in the array of V previously created which did not contain boundary conditions, and perform the approximation described by equation (3).The algorithm visits every point in V (not contained in the plates) and then starts again, over and over until the values converge to a sensible error I labelled as 'rtol', and then saves that array as 'array V':

```
calculatednodes= np.load("C:/Users/User/Desktop/calculated nodes.npy")
for a in range(Nmax):
    for [i,j,l] in calculatednodes:
        Vc[i,j,l]=(V[i+1,j,l]+V[i-1,j,l]+V[i,j-1,l]+V[i,j+1,l]+V[i,j,l-1]+V[i,
    j,l+1])/6
        if abs(V[i,j,l]-Vc[i,j,l]) > rtol:
            V[i,j,l]=Vc[i,j,l]

np.save("array V",V)
```

Results

After the algorithm is done, we end up with a matrix of size m,n,k (the length of the x,y and z dimension respectively) containing all the values for the electric potential in the domain we chose around the capacitor. I chose to extract the values from V from different planes to plot different diagrams.

I will start by plotting the potential along the x-axis.This is, along the axis that crosses the two plates at their center when y=0 and z=0.I will also plot the potential along the sides of the plates, which is the potential along the x-axis at points y=2.5 cm and z=5 cm

```
plt.figure()
plt.plot(x,V[:,75,150],'Black',label='Along center')
plt.plot(x,V[:,100,150],'g--',label='Along side')
plt.plot(x,V[:,75,200],'r',alpha=0.7,label='Along top')
plt.legend(loc='upper right')
plt.xlabel('x(cm)')
plt.ylabel('V(V)')
plt.grid(alpha=0.2)
plt.axvspan(-0.5,0.5,alpha=0.2)
```

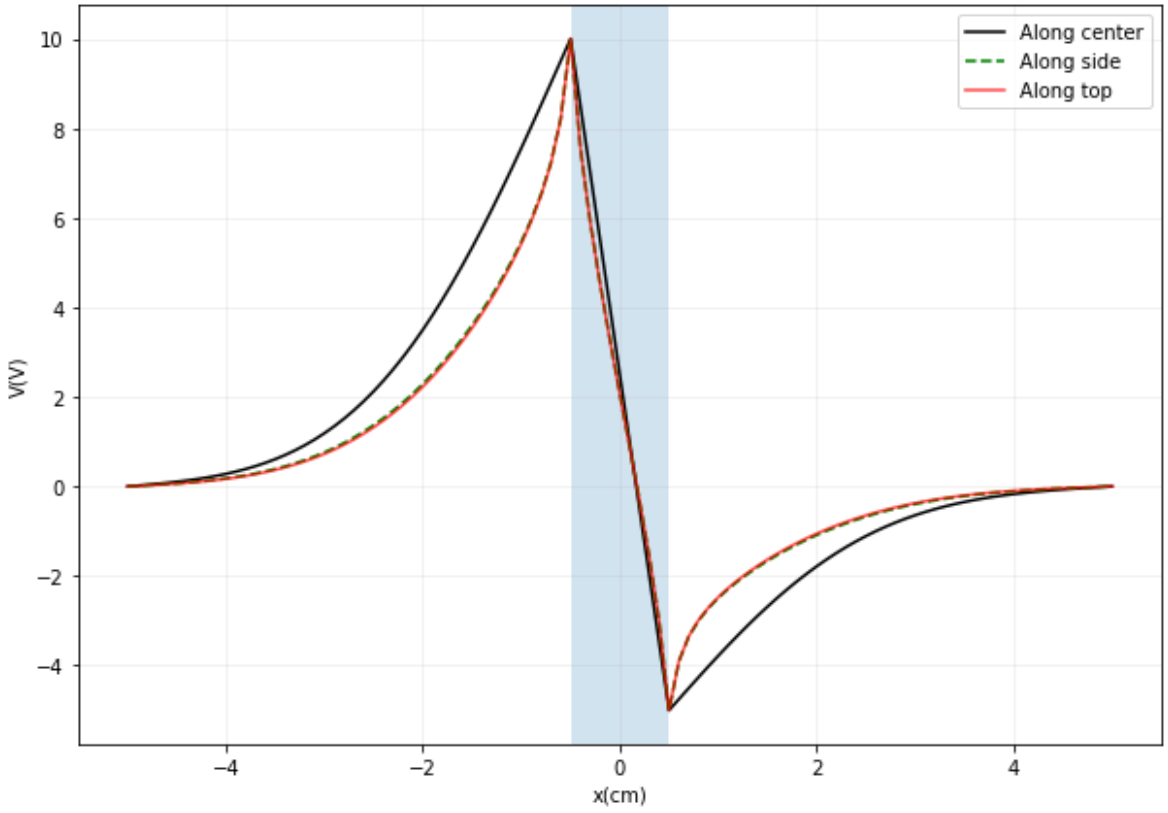

Figure 1: Potential along the x-axis at different positions of the plates.The shaded region indicates the space between the plates.

as the figure suggests, the potential along the top and the side of the plates are virtually identical, differing only with the potential along the center.

It is also appropriate to compare these results with the potential for a **infinite** parallel-plate capacitor.The potential for this arrangement is:

$$V(x) = \left\{ \begin{array}{ll} V_2 + (V_1 - V_2) \cdot (\frac{d-2x}{2d}), & -d/2 < x < d/2 \\ V_1, & x < -d/2 \\ V_2, & x > d/2 \end{array} \right\} \tag{4}$$

Where V_1, V_2 are the potential of the plates and d the space between them.

If I now add this function to the previous plot:

```
d=1
v1=10
v2=-5
h=0.025
x1=np.arange(-0.5,0.5+h,h)
theoretical = lambda x: v2+(v1-v2)*((d-2*x)/(2*d))
plt.hlines(y=10, xmin=-5, xmax=-0.5, linewidth=2, color='Purple',label='
    Theoretical')
plt.plot(x1,theoretical(x1),color='Purple')
plt.hlines(y=-5, xmin=0.5, xmax=5, linewidth=2, color='Purple')
plt.legend(loc='upper right')
```

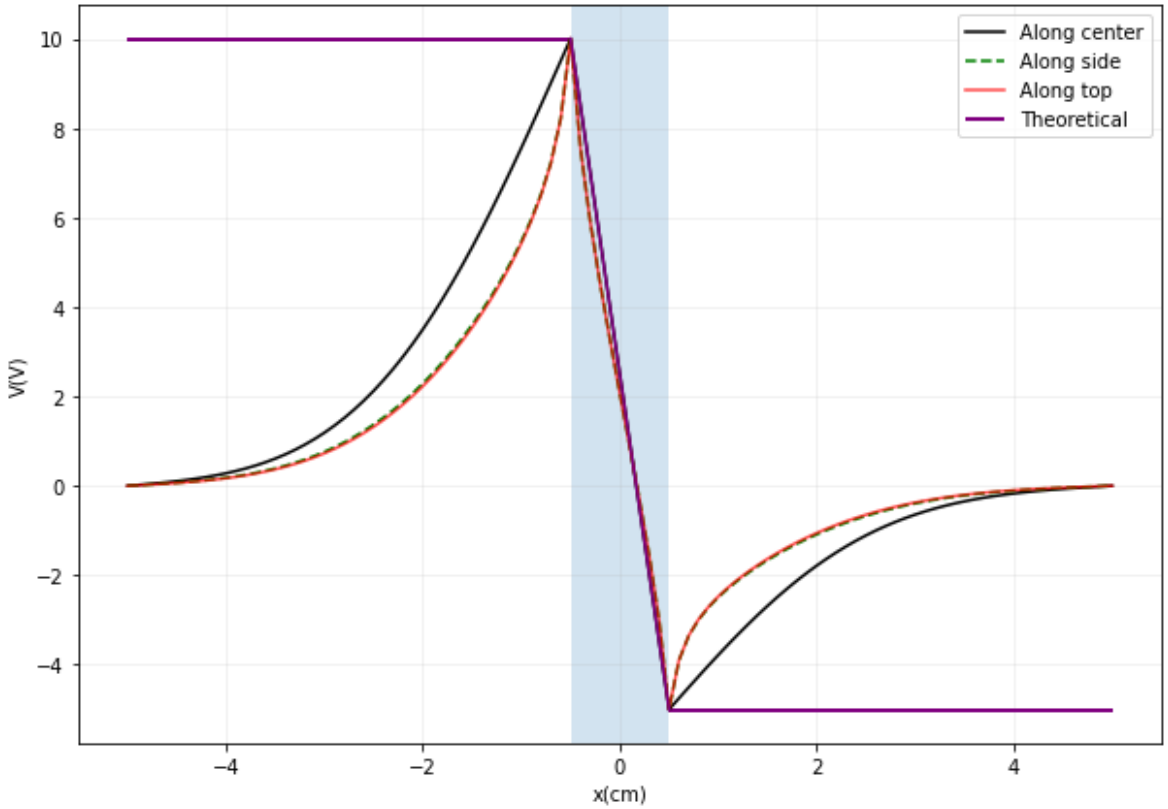

Figure 2: Potential across the x-axis.The purple line indicates the Theoretical potential for **infinite** plates

We can see how the difference becomes more pronounced as we go further from the center of the plates, the center portion being almost identical (region in shaded blue). **However**, we

must bear in mind that in the numerical simulation we placed a **conducting** box around the plates, where we set the potential to zero and due to the necessity of the potential to be continuous everywhere, produces the gradient taking the potential to zero at both ends of the space that we can appreciate . This situation is **not** taken in count when calculating the theoretical potential. Thus, it is only appropriate to compare both potentials at the central region. To give a bit more insight in this argument, I also plotted the difference between the numerically-obtained potential and the value of the theoretical potential at the center of the plates, in the following manner:

$$Difference = V_{num}(0, y, 0) - V_{inf}(0)$$
$$Percentage\ difference = \frac{Difference}{V_{inf}(0)} \times 100$$

Where the value $V_{inf}(0)$ is chosen to emphasize that this is only a sensible comparison at the center region, as stated above.

About the coding I used for this part: I produced two lists containing the values of the difference and percentage difference, as defined above, by iterating the equation through all the points in y. Then, plotted these vectors against the vector y.The percentage difference is shown in red and its scale is the one displayed by the right y-axis. The difference is shown in black and belongs to the left y-axis.

```
#%%
difference = []
for t in range(0,len(y)):
     difference.append( V[50,t,150]-theoretical(0))
#%%
percdifference = []
for g in range(0,len(y)):
     percdifference.append( -(difference[g]/theoretical(0))*100)
#%%
fig, ax1 = plt.subplots(figsize=(10, 7))
ax2 = ax1.twinx()

ax1.plot(y, difference,color='Black',label='Difference')
ax2.plot(y,percdifference,color='Red',label='Percentage difference');
ax1.legend(loc='center left')
ax2.legend(loc='center right')
ax1.set_xlabel('y(cm)')
ax1.set_ylabel('V(V)')
ax2.set_ylabel('%')
plt.axvspan(-2.5,2.5,alpha=0.2)
```

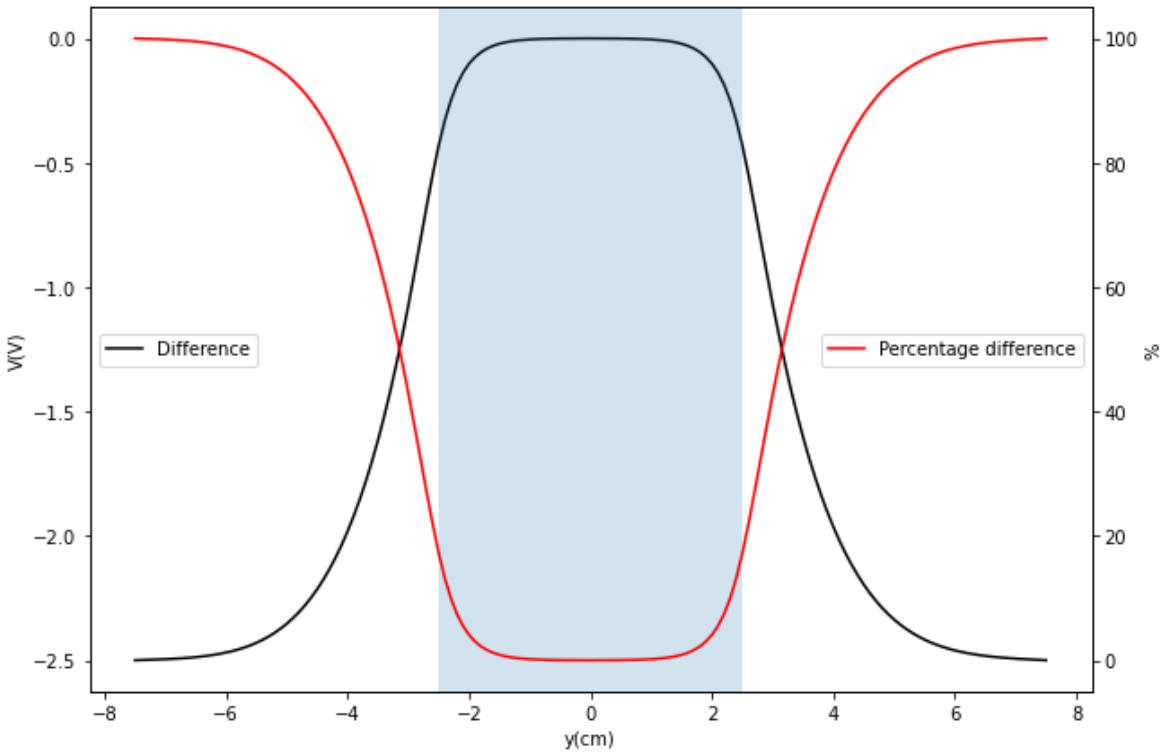

Figure 3: Difference and percentage difference between the potentials for finite and infinite plates.

One might also be interested in quantitatively analyze this graph.For instance, where is the percentage difference greater than 10%? **(QUESTION 2i)**

To find this answer, I made a loop that would call all the values of the percentage difference obtained before (starting from y=0, or j=75 in the numerical grid) and compare them with number 10. The moment one of them is greater than 10, the loop stops and its place in the array recorded:

```
for r in range(len(y)//2,len(y)):
    if percdifference[r] > 10:
        print('potential is greater than 10% from point '+ str(y[r])+'on')
        break
```

```
>>> 'potential is greater than 10% from point 2.399999999999965 on '
```

Which outputs that the answer to this question is that the percentage difference is greater than 10% everywhere outside the interval $-2.4 < y < 2.4$.
One might also want to know what is the percentage difference at the edge of the plates **(QUESTION 2ii)** (since its where the fringing of the field starts to be noticeable).We can use an altered version of the code above:

```
for w in range(len(y)//2,len(y)):
    if y[w]-2.5 <0.01:
        print('percentage difference at the edge of the plates is '+ str(
    percdifference[r]))
        break
```

```
>>>'percentage difference at the edge of the plates is 13.229998791495614'
```

Thus as we can see the percentage difference at the edge of the plates is 13.23%.

Finally, one can also plot a contour plot showing the equipotential lines of the field around the parallel-plate capacitor.These equipotential curves are curves such that the potential throughout them is a constant value.I will plot the equipotentials for voltages from -4V to 9V

```
plt.figure(figsize=(10,7))
CS = plt.contour(Y,X,V[:,:,150],levels=[-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9],cmap=
    'inferno')
h = CS.collections
l = [f'{a:.1f}' for a in CS.levels]
plt.legend(h,l)
#plt.clabel(c, inline=1, fontsize=10)
plt.xlabel('x(cm)')
plt.ylabel('y(cm)')
plt.grid(alpha=0.2)
#plt.show()
```

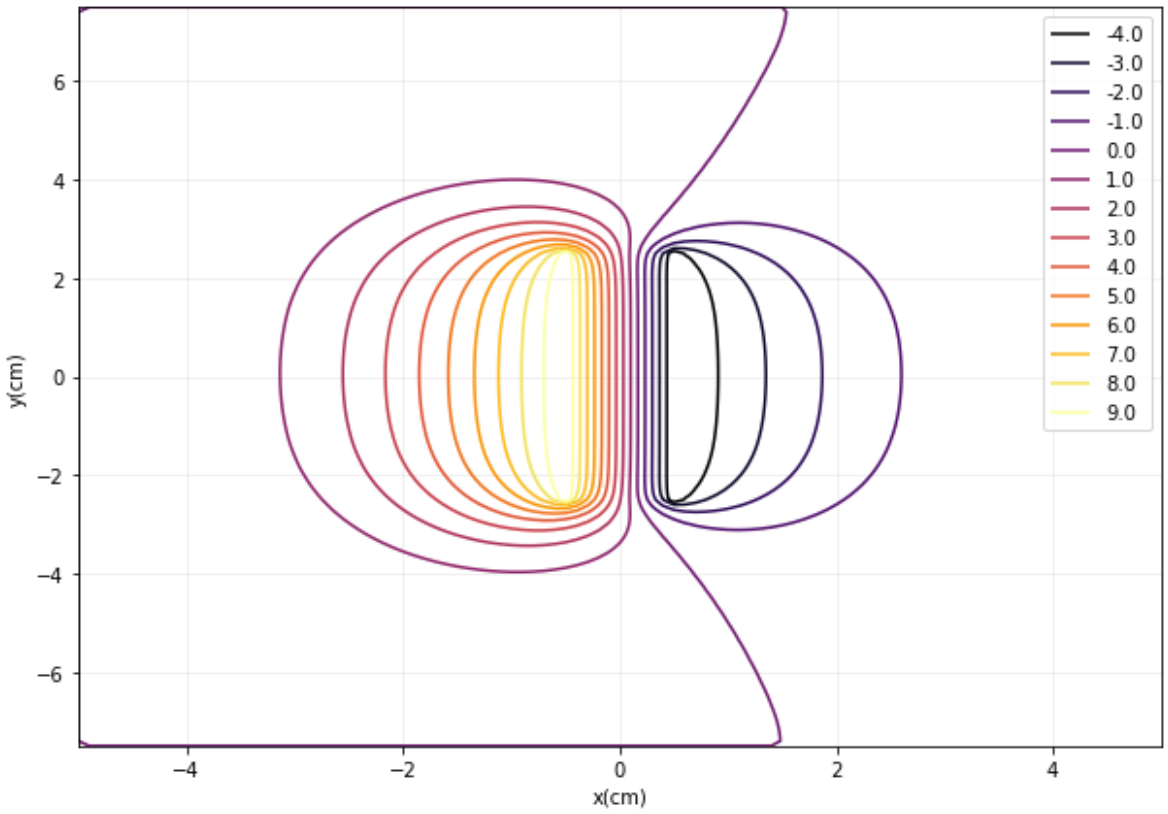

Figure 4: equipotential curves in the XY plane.Note that the plates are located around points (-0.5,0) and (0.5,0)

One of the definitions of the electric field is such that it is described as being the negative gradient of the Coulomb's potential:

$$E = -\nabla V$$

Thus we can see that the Electric field will be strongest in the region where the potential changes faster, which happens to be between the plates as the equipotentials are closer together (eg,: the potential goes from -4V to 9V in a shorter distance) **(Question 3i)**
Following the same reasoning one can say that the electric field in the left side of the capacitor (at x<-0.5) is greater than at the right side (x>0.5) since we have more equipotentials per unit distance, *the potential changes faster there.* A reason for this is that our boundary conditions establish that the potential everywhere at the box surrounding the capacitor (which is symmetric) is zero, and having the left plate a bigger absolute potential than the right (10V vs 5V) the potential at the left must decrease at a faster rate to satisfy the boundary conditions. **(Question 3ii)**.

One might be interested in the equipotential curve for V=0, since something seems off about it -it stretches to the positive x-axis rather than staying parallel to the y-axis. The only explanation I can think about is got to do with the electric field of the left side being assymetrical outside the region between the plates.Thinking of the field as the gradient of the potential, and recalling that the field decays with distance squared, as we get further away from the plates, the field decays faster than the potential, so the further away, the longer it takes for the potential to reach 0 with respect to distance, accentuated by the fact that the equipotential for V=0 not being perfectly centered between the plates, but slightly off to the right (consequence of the left plate having a greater potential magnitude) **(Question 3iv)**

Bearing this in mind, I will try to illustrate how the electric field around this parallel plate capacitor might look like **(Question 3iii)**

As far as multivariable calculus goes, multivariable scalar functions (such as the Coulomb's potential) have a notion of **directional derivatives**, where the gradient of these kind of functions is a vector which varies with the direction where it points at any given point.This directional derivative will be maximum in the orientation where it matches with the gradient of the function, and it is the direction perpendicular to the equipotential curves (i.e.: where the gradient is zero). Thus, the electric field, being a gradient (vector) field, will always point in a direction perpendicular to the equipotential curves of the potential:

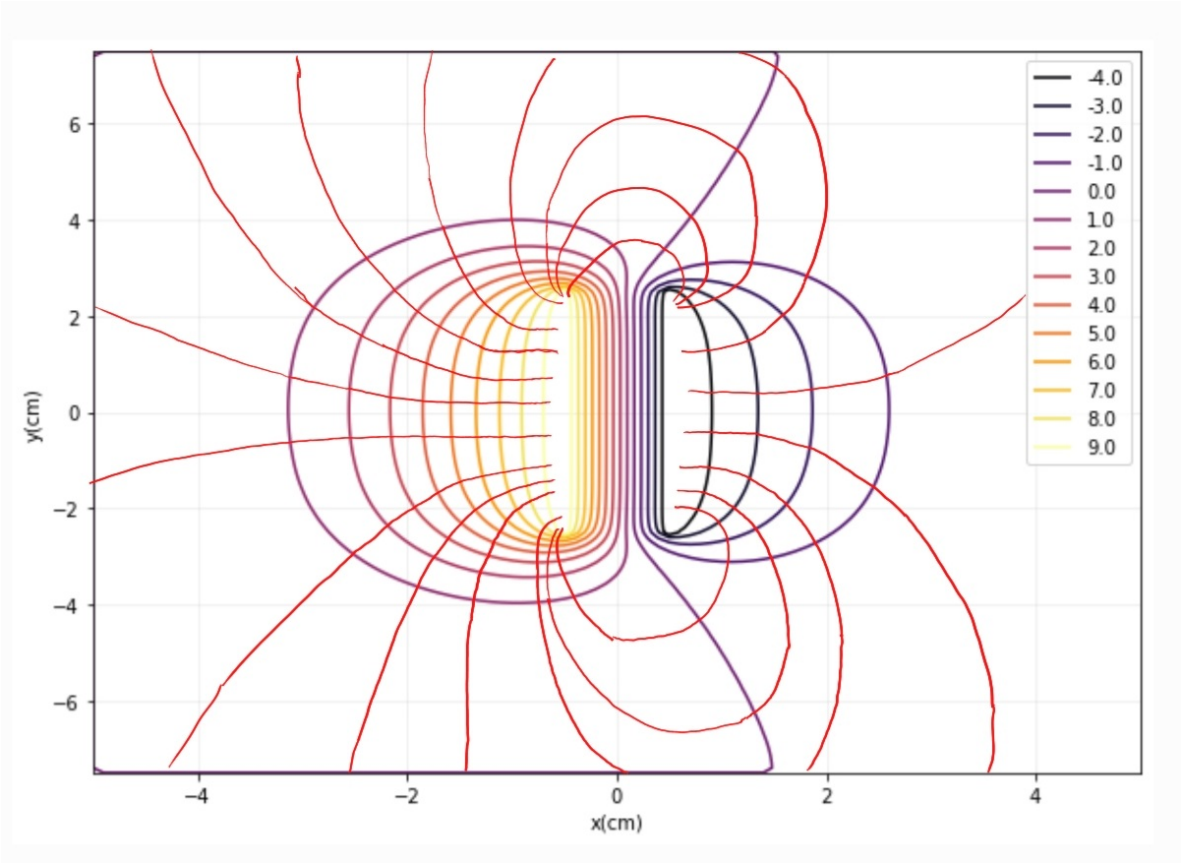

Figure 5: equipotential curves in the XY plane with the resulting electric field in red.Note that the plates are located around points (-0.5,0) and (0.5,0)

This would be roughly how the field looks, and it somehow hints the argument made earlier about the overall field not being symmetrical with respect to x=0. I should really state that the **direction** of these lines goes from the positively charged plate (the left) to the negatively charged plate (the right). The arrows to signal its direction were left out for clarity.