Homework 7

Cassie Metzger

November 7, 2023

1 2D Exercise

This problem can be solved numerically in `python`. It can be found in the repository under the number `problem1.ipynb`.
This problem can be solved using Laplace's equation ($\nabla^2 V = 0$) because there is no charge within the desired region. Using the boundary conditions, we can find a function $V(x, y)$. In this case, V is 0 along three of the four sides. But, along the fourth side, the electric potential behaviors according to $V(x, y = 0) = V_0 \sin\left(\frac{2\pi x}{a}\right)$

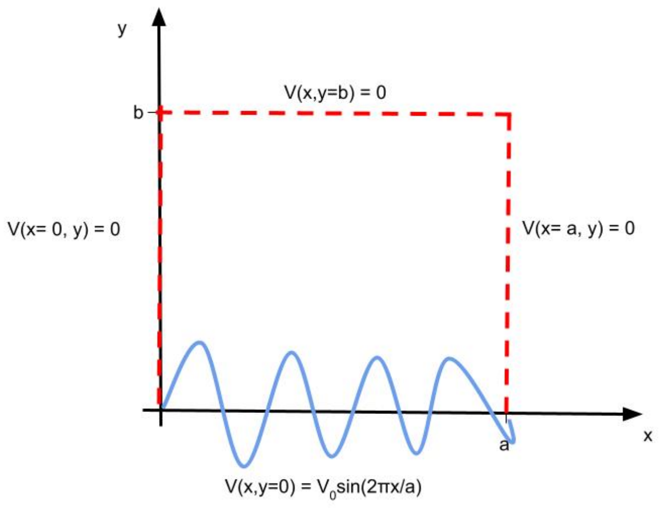


Figure 1: Depiction of the boundary conditions

The boundary conditions are written in the code:

```
V[:,0]= 0 #x = 0, y
V[:,-1]=0 #x = a, y
V[-1,:]=0 # x, y = b
V[0,:] = V0 * np.sin(2*np.pi * (x[0, :])/a) # x, y = 0
```

Now, in order to solve this problem, we can use the *finite-difference method.* The finite-difference method essentially consists of a grid. Each point on the grid if Δx away from its horizontal neighbors and Δy away from its vertical neighbors. V at each point can be found by taking the average of its surrounding points. A graphical depiction of this can be found in Figure [2]

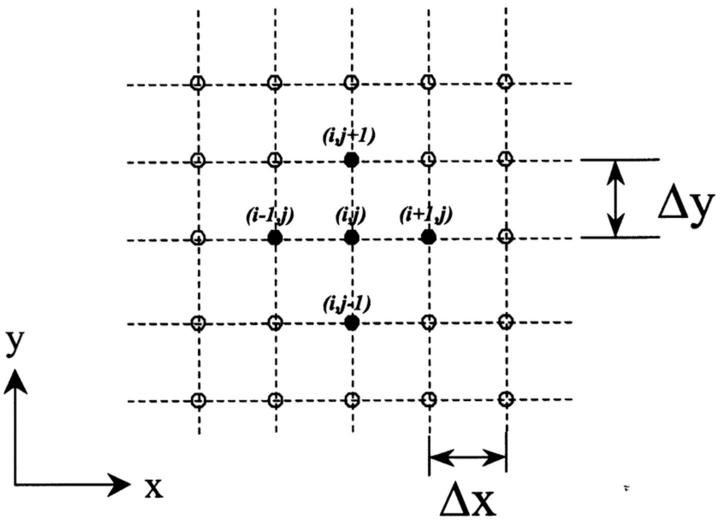


Figure 2: A depiction of the spatial grid that the finite difference method integrates over.

In this case Δx and Δy are chosen to be the same. So, Δ is initially chosen to be $\frac{1}{50}$ (represented as `dx=1.0/50` in the code). The interval for x in this problem is $[0, a]$ and the interval for y is $[0, b]$. We can construct a range of x-values (`nx`) and a range of y-values (`ny`).

```
nx=int((xmax-xmin)/dx)
ny=int((ymax-ymin)/dx)
```

These values are then used to construct the grid:

```
x0=np.linspace(xmin,xmax,nx+1)
y0=np.linspace(ymin,ymax,ny+1)
x,y=np.meshgrid(x0,y0)
```

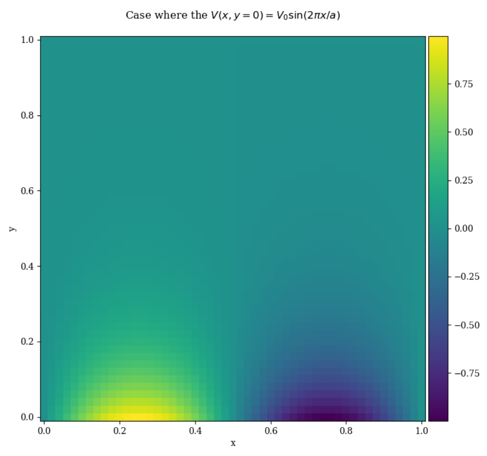
After the grid is constructed, all values are set to 0 (`V=np.zeros_like(x)`). Then, the boundary conditions [Code 1] are applied.
To carry out the finite-difference method, a tolerance must be defined. The code will loop through the algorithm until the tolerance is reached. Mathematically, the finite difference method can be defined as the following average.

$$V_{i,j}^{n+1} = \frac{1}{4}(V_{i-1,j}^{n} + V_{i+1,j}^{n} + V_{i,j-1}^{n} + V_{i,j+1}^{n}) \tag{1}$$

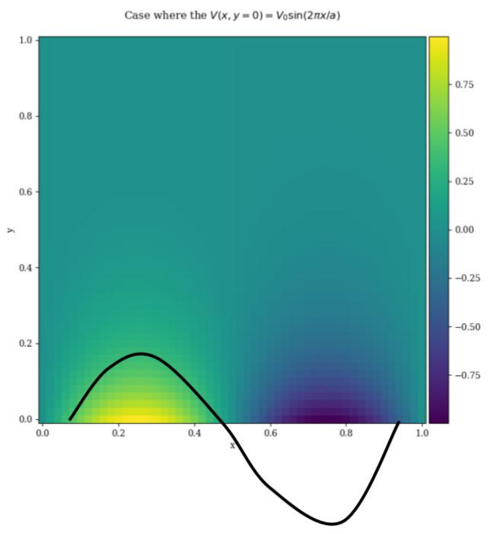
In the code, a tolerance (`eps`) is defined as `eps=1.0e-6`. Equation [1] is then used to calculate initial values for the grid. This initial value (`resid`) is then used as a comparison as the code loops through and recalculates the values for each grid point until `resid/resid0<=eps`. This is shown:

```
# Tolerance for convergence
eps=1.0e-6

# Carry out the first round of the averaging scheme and calculate the residual
V1=np.copy(V) # a temporary array to store the updated values of V
V1[1:-1,1:-1]=0.25*(V[0:-2,1:-1]+V[2:,1:-1]+V[1:-1,0:-2]+V[1:-1,2:]) # carry out t
resid=np.sum(np.fabs(V1-V)) # calculate the initial residual
resid0=resid
V=np.copy(V1) # Put updated values into the array V

start=time.time()
it=0
while (resid/resid0>eps): # Loop criterion: we use relative residual here
    V1[1:-1,1:-1]=0.25*(V[0:-2,1:-1]+V[2:,1:-1]+V[1:-1,0:-2]+V[1:-1,2:])
    resid=np.sum(np.fabs(V1-V))
    V=np.copy(V1)
    it=it+1
end=time.time()
print("Number of iterations:",it)
print("The time spent is",end-start,"s")
```

To view the solution, we can plot the x vs. y grid with a color scale depicting the value of V at each point along the grid.



(a) The electric potential over a grid with $\Delta = \frac{1}{50}$ and tolerance $= 1 \times 10^{-6}$



(b) The electric potential graph with a sine curve overlaid to depict how the boundary condition at $V(x, y = 0)$ resembles a sine curve with an amplitude of 0.75.

This graph doesn't represent the error that is accrued as the finite difference method is run. To understand the error, the problem can be solved analytically. The analytical solution can then be compared to the numerical solution. The analytical solution is as

follows.
From the boundary conditions, $V = 0$ at the `xmax` and `xmin`, therefore, the general solution to x can be written as

$$X(x) = A_1 \sin kx + A_2 \cos kx \tag{2}$$

Since $V \neq 0$ at both y boundaries, the general solution to y can be written as

$$Y(y) = B_1 e^{ky} + B_2 e^{-ky} \tag{3}$$

Using the boundary conditions:

(i) $V(x, y = 0) = V_0 \sin(2\pi \frac{x}{x})$

(ii) $V(x, y = b) = 0$

(iii) $V(x = 0, y) = 0$

(iv) $V(x = a, y) = 0$

The general solutions can be simplified
(iii) $0 = A_1 \sin(0) + A_2 \cos(0)$
$A_2 = 0$
(iv) $0 = A_2 \sin ka$
$\sin ka = 0$
$ka = \pi n$
$k = \frac{\pi n}{a}$
(ii) $0 = B_1 e^{kb} + B_2 e^{-kb}$
$\frac{B_1 e^{kb}}{B_2 e^{-kb}} = -\frac{B_2 e^{-kb}}{B_2 e^{-kb}}$
$-B_1 e^{2kb} = B_2$
Therefore, $V(x, y)$ can be written as

$$V(x, y) = \sum_{n=1}^{\infty} C \sin(kx)(e^{ky} - e^{k(2b-y)}) \tag{4}$$

Where $C_n = A_1 B_1$
The first boundary condition can be applied:
(i) $V_0 \sin(2\pi \frac{x}{a}) = \sum_{n=1}^{\infty} C_n \sin(kx)(e^0 - e^{k(0-2b)})$
To simplify, I set $\gamma = 1 - e^{-2kb}$ and plug in $k = \frac{\pi n}{a}$
Then, $V_0 \sin(2\pi(\frac{x}{a})) = \sum_{n=1}^{\infty} C_n \sin(\frac{\pi n x}{a})\gamma$
By applying Fourier's trick:

$$V_0 \int_0^a \sin(2\pi(\frac{x}{a})) \sin(\frac{\pi n' x}{a}) dx = \sum_{n=1}^{\infty} C_n \gamma \int_0^a \sin(\frac{\pi n x}{a}) \sin(\frac{\pi n' x}{a}) dx \tag{5}$$

$\int_0^a \sin(\frac{\pi n x}{a}) \sin(\frac{\pi n' x}{a}) dx$ equals 0 if $n' \neq n$ and $\frac{a}{2}$ if $n' = n$
The same rule can be applied to the right-hand side of the equation. $\sin(2\pi(\frac{x}{a})) = \sin(\frac{\pi n' x}{a})$ if $n = 2$. So, we can say that $n = 2 = n'$.

$$V_0(\frac{a}{2}) = C\gamma(\frac{a}{2}) \tag{6}$$

$$C = \frac{V_0}{1 - e^{\frac{4\pi b}{a}}} \tag{7}$$

Thus, we can write

$$V(x,y) = \frac{V_0}{1 - e^{\frac{4\pi b}{a}}} \sin\left(\frac{2\pi x}{a}\right)\left(e^{\frac{2\pi y}{a}} - e^{\frac{2\pi}{a}(2b-y)}\right) \tag{8}$$

This solution is then plotted so that it can be compared to the numerical solution. The following code constructs a grid of x and y values and calculates $V_{analytical}$ for each point.

```
xa,ya=np.meshgrid(x0,y0)
V_ana = np.zeros_like(xa)
range_x = np.arange(0, nx, 1)
range_y = np.arange(0, ny, 1)
for i in range_x:
    for j in range_y:
        V_ana[i,j] = V0/(1- np.exp((4*np.pi*b)/a))*
        np.sin((2*np.pi*xa[i,j])/a)*
        (np.exp((2*np.pi*ya[i,j])/a)-
        np.exp(np.pi*(4*b -2*ya[i,j])/a))
```

This produces the following graph:

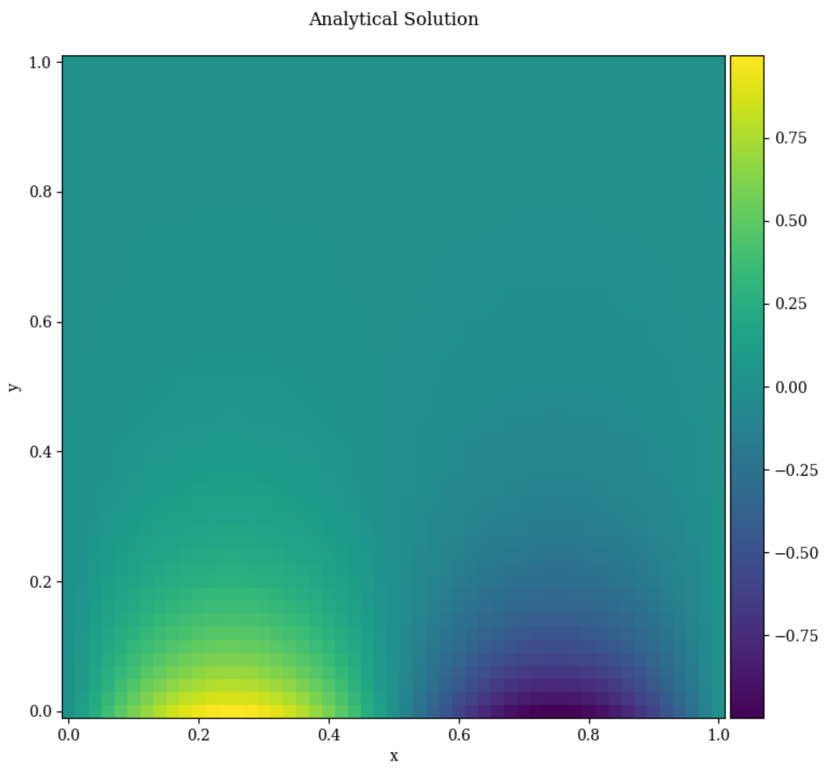


Figure 3: The analytical solution to the Laplace equation given specific boundary conditions

Both solutions seem roughly on target since they produce similar graphs. Therefore, the error of the numerical solution can be checked. The error is calculated with the following formula:

$$error = |V_{numerical} - V_{analytical}| \tag{9}$$

The error is calculated using the following bit of code where `V_err` represents *error*.

```
xe, ye = np.meshgrid(x0, y0)
V_err = np.zeros_like(x)
range_x = np.arange(0, nx, 1)
range_y = np.arange(0, ny, 1)
for i in range_x:
    for j in range_y:
        V_err[i,j] = np.absolute(V_ana[i,j] - V[i,j])
```

The error is then plotted on the same x and y grid as the earlier functions.

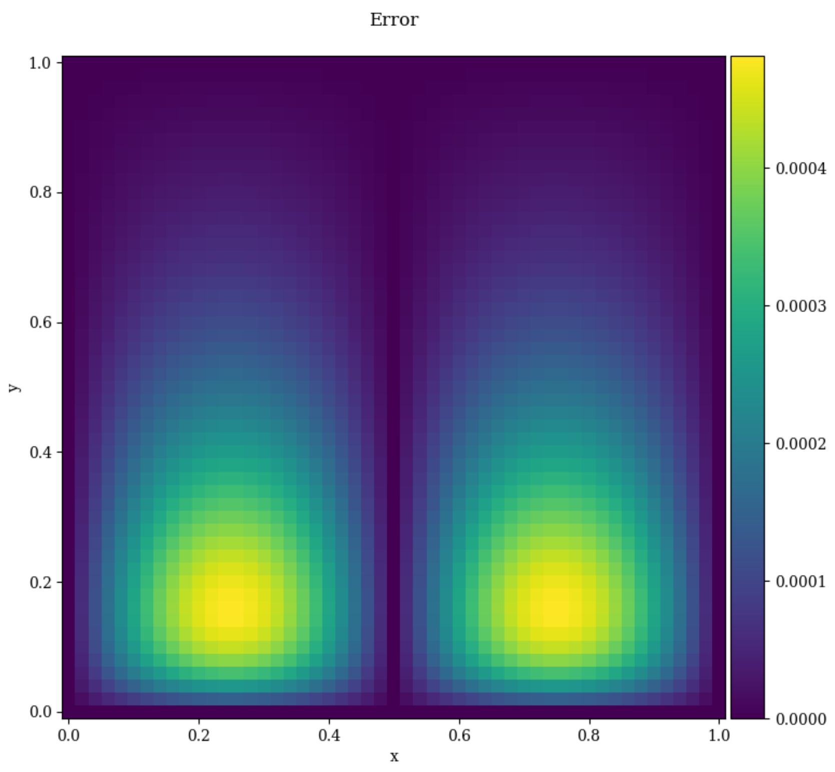
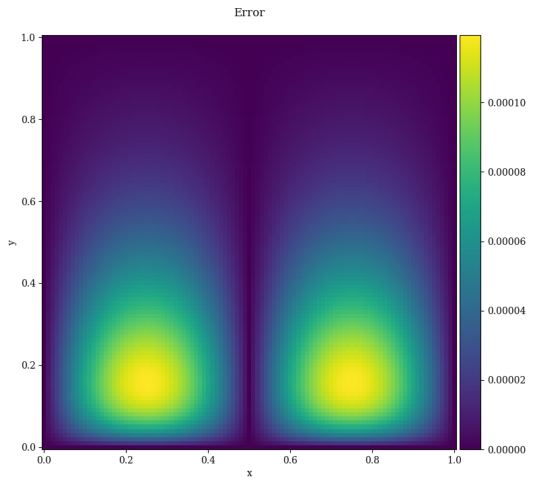
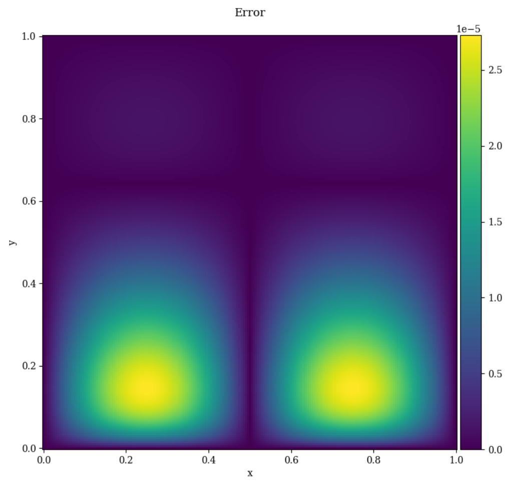


Figure 4: The error when the step size, dx, $= \frac{1}{50}$ and tolerance $= 1 \times 10^{-6}$

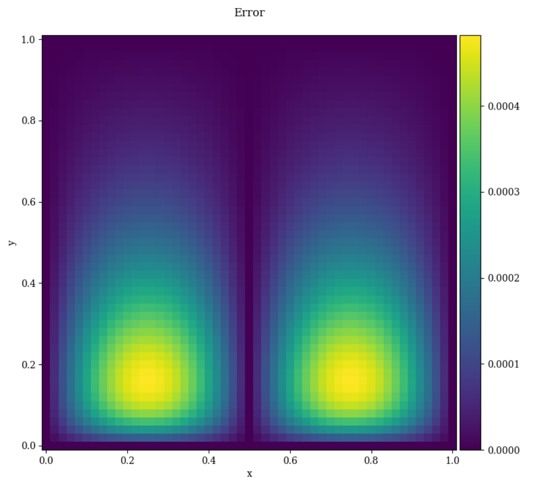
The error can be decreased by lowering the tolerance and/or by making the step size smaller.



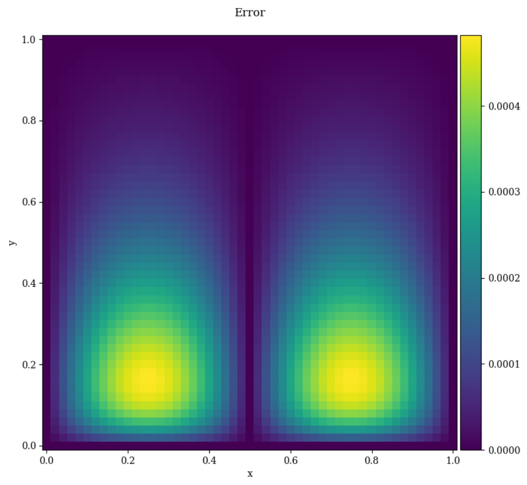
(a) The error when the step size is lowered by a factor of 2, but the step size remains at `dx` $= \frac{1}{50}$



(b) The error when the step size is lowered by a factor of 4, but the tolerance remain at 1×10^{-6}



(c) The error when the tolerance is lowered by a factor of $\frac{1}{10}$, but the step size remains at `dx` $= \frac{1}{50}$



(d) The error when the tolerance is lowered by a factor of $(\frac{1}{10})^2$, but the step size remains at `dx` $= \frac{1}{50}$

However, it appears that lowering the tolerance below 1×10^{-6} does very little to affect the error. If the tolerance were greatly increased, I expect that there would be greater variation, but it doesn't appear that the image can be refined much below an error of 1×10^{-6}. On the other hand, the step size greatly affects the resolution. Larger step sizes produce more pixelated images. This is shown in the fact that by lowering the step size by just $\frac{1}{2}$, the maximum error decreases from 4×10^{-4} to 1.0×10^{-4}. However, there is a limit to how low the step size can go since decreasing the step size has a much greater effect on the run time than decreasing the tolerance does. The correlation between the step size and the error can be seen in Figure [5]

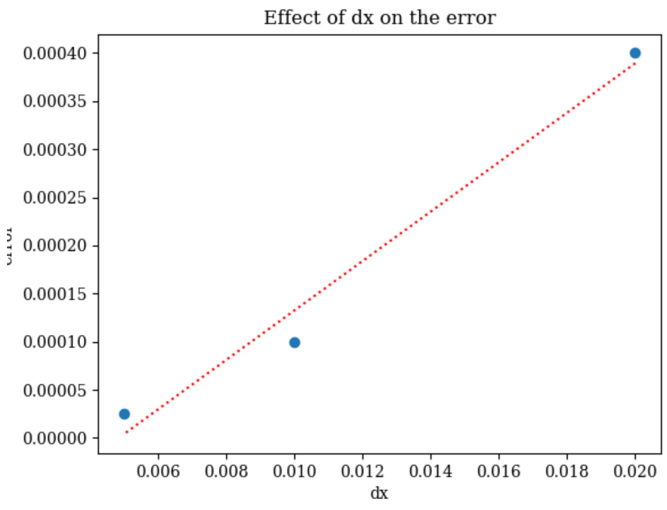


Figure 5: The correlation between the step size and the corresponding error, plotted with a simple linear fit.

It's interesting to note that lowering both the step size and the tolerance does not necessarily lower the error.
Finally, even though this problem is technically 2D as its boundary conditions only pertain to the x and y directions, the solution can be plotted in 3D. When plotted in 3D, the sine curve that the analytical solution portrays can be clearly seen.

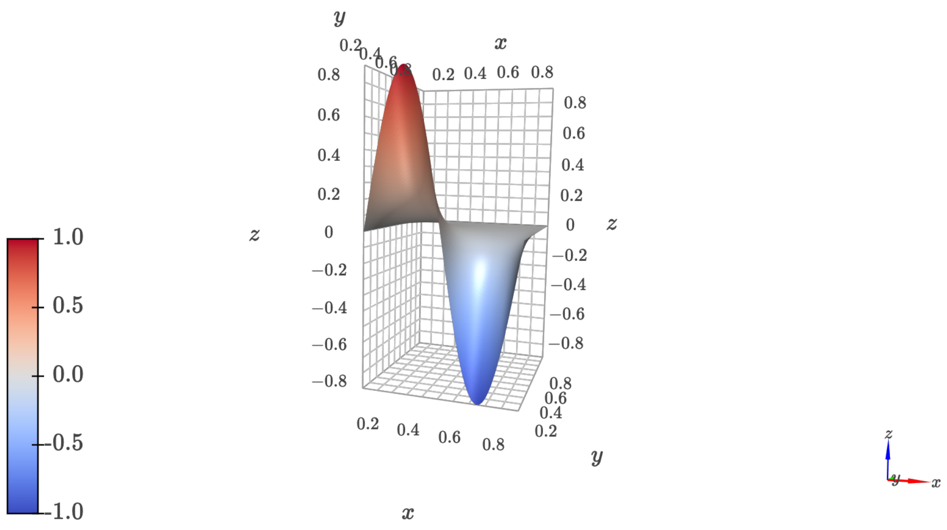


Figure 6: The solution plotted in 3D

1.1 When $V(x, y = 0) = V_0$

I repeated the same procedure but replaced the boundary condition $V(x, y = 0) = V_0 \sin\left(\frac{2\pi x}{a}\right)$ with $V(x, y = 0) = V_0$. Since the error had already been estimated, I chose to begin with a step size of $\frac{1}{100}$ in order to reduce the error. The solution is computed numerically in the file `problem2.ipynb`. After re-implementing the code, I produced this graph:

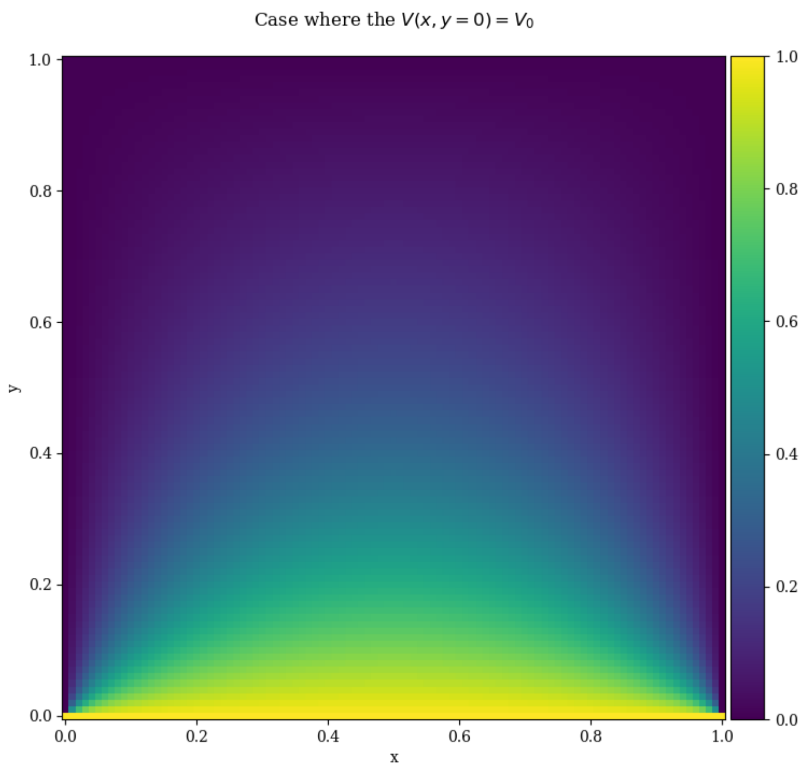


Figure 7: A 2D plot of the electric potential in the second case.

This result makes sense! In case 1, the lower boundary condition was given as a sine function, therefore, the graph produced two curves. In this case, the lower boundary condition is given as V_0 which is assigned to be 1. So, the boundary is constant and equal to V_0.

A 3D graph of this case can also be produced as seen in Figure [8].

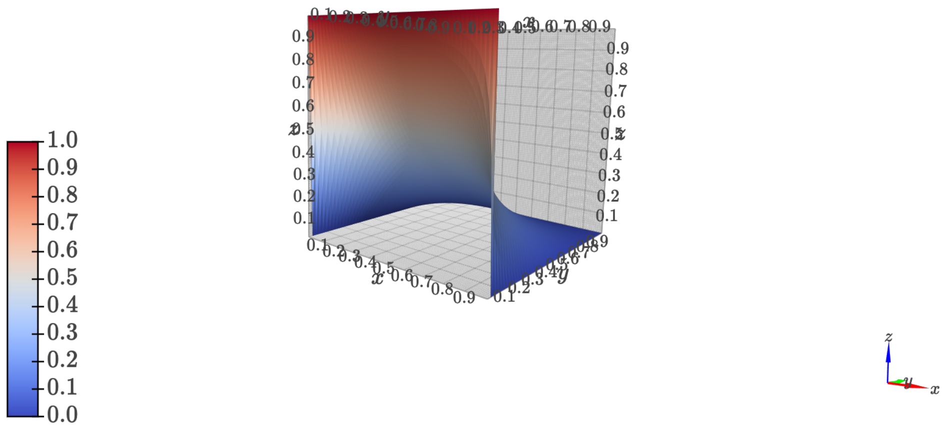


Figure 8: A 3D graph of Case 2

Again, this graph makes sense. The potential appears to increase in magnitude as it moves across the x and y grid, reaching a maximum potential that is equivalent to V_0

2 3D Exercise

The procedure is very similar in 3D. Again, the aim is to solve the Laplace equation, so the region of interest must have no enclosed charge. Of course, one important difference is that another dimension is added to the problem. Therefore, the aim is to solve the equation: $\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial V^2}{\partial z^2} = 0$. This is implemented in the file `problem3.ipynb`.
First, to get an idea of the problem, we can visualize the boundary conditions. x varies between 0 and a, y varies between 0 and b, and z varies between 0 and c. The potential at the boundaries is 0 in all cases except for $V(x = a, y, z) = V_1$ and $V(x = 0, y, z) = V_0 \frac{y}{b}(1 - \frac{y}{b})\frac{z}{c}(1 - \frac{z}{c})$.

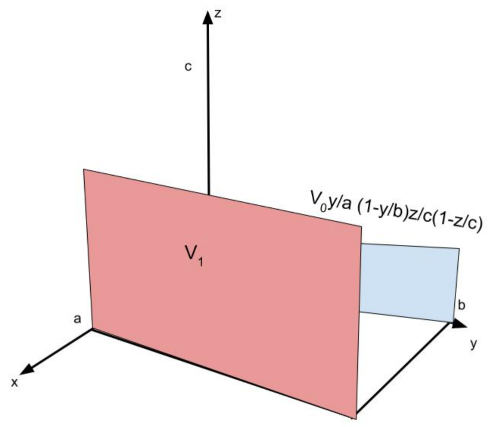


Figure 9: A description of the 3D boundary conditions

There are more boundary conditions now since the boundaries depend on x, y and z. The boundary conditions are written in the code:

```
# Apply boundary conditions
V[:,:,0]=V0*(y[:,:,0]/b)* (1-(y[:,:,0]/b))
*(z[:,:,0]/c)*(1-(z[:,:,0]/c)) # x=0, y, z
V[:,:,-1]=V1 # x=a, y, z
V[:,0,:]=0 # x, y=0, z
V[:,-1,:]=0 # x, y=b, z
V[0,:,:]=0 # x, y, z=0
V[-1,:,:]=0 # x, y, z=c
```

In order to go about solving the 3D problem, we can add a third dimension to Equation (1). Now, we're taking the average of 6 points, in this case, we can set all of the points to be equally spaced from each other. This can be seen below:

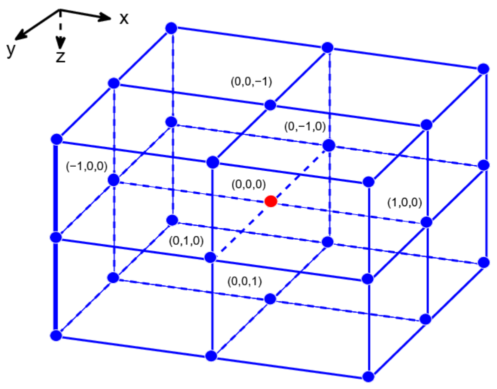
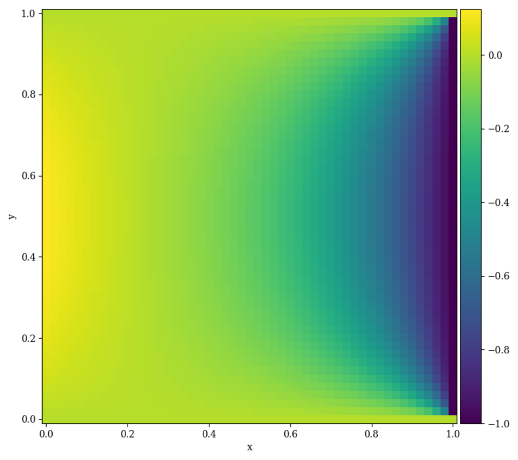


Figure 10: The 3D spatial grid that finite difference method integrates over

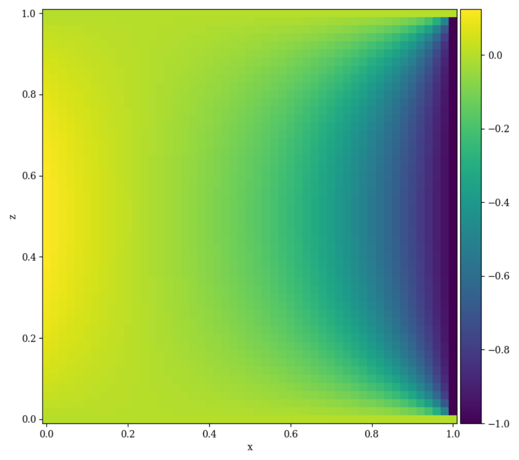
Therefore, just as before, V at each point can be calculated by the average of the surrounding points. In 3 dimensions, two additional terms are added to the average.

$$V_{i,j,k}^{n+1} = \frac{1}{6}(V_{i-1,j}^n + V_{i+1,j}^n + V_{i,j-1}^n + V_{i,j+1}^n V_{i,j,k-1}^n + V_{i,j,k+1}^n) \tag{10}$$

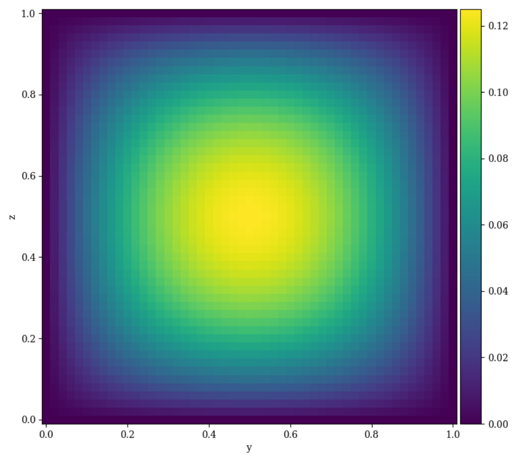
The code to implement this is exactly the same, except for the fact that two additional terms are now included in $V^{n+1}_{i,j,k}$.
Given that boundary conditions are in 3 dimensions, a 2D plot of V does not completely depict the situation. Therefore, three $2D$ plots can be made to depict that boundary conditions in the xy-plane, the yz-plane, and the yz-plane.



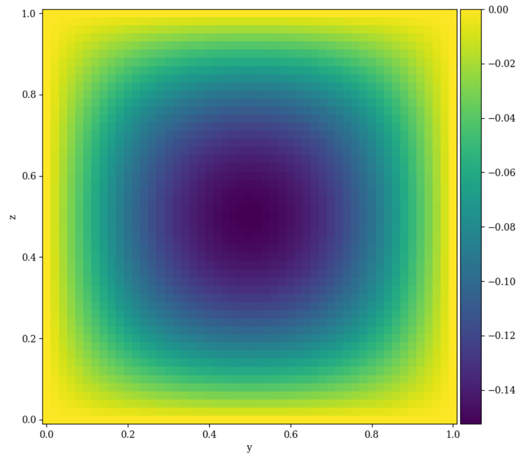
(a) The potential in the xy-plane when $z = 0.5$



(b) The potential in the xz-plane when $y = 0.5$



(c) The potential in the yz-plane when $x = 0$



(d) The potential in the yz-plane when $x = 0.5$

It should be noted the step size `dx` (distance between points on the grid) was chosen to be $\frac{1}{50}$. Even though it was shown in the 2D problem that the step size $\frac{1}{100}$ minimized the error without greatly increasing the run time, this was not true in the 3D case. The run time greatly increased when the step size was decreased by a factor of $\frac{1}{2}$. In addition, the tolerance (`eps`) was chosen to be 1×10^{-6}.

Now, these plots don't seem to make much sense individually. But, one thing of note is that the xy and the yz planes appear very similarly in terms of their potential while the yz plane is vastly different. To explore this, we can plot the potential in 3D, allowing all of the plots to come together.

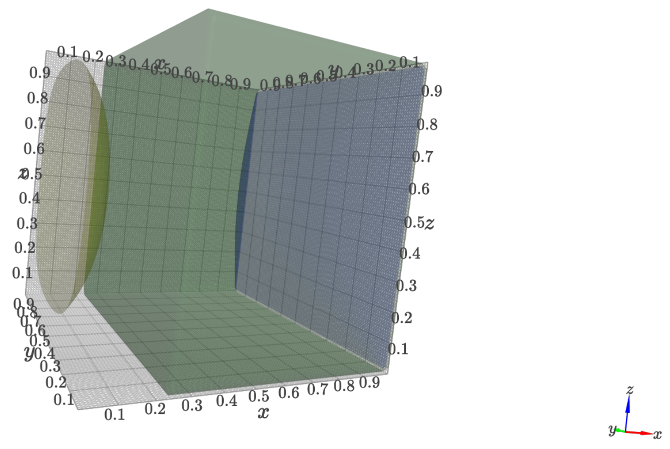
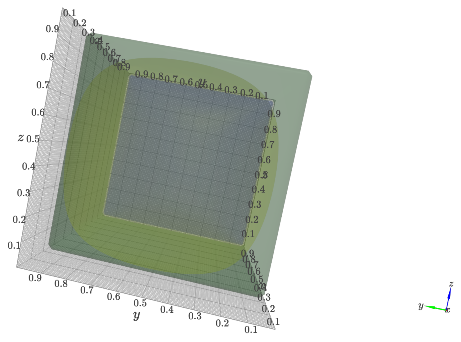
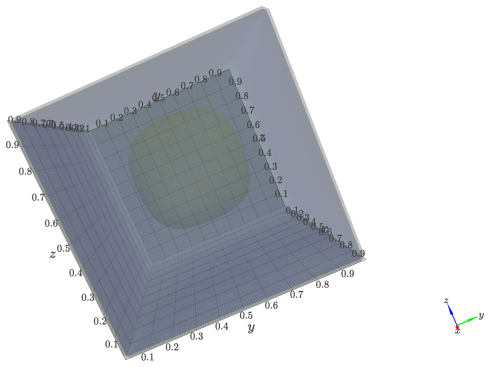


Figure 11: The 3D exercise plotted in 3D

This plot is created from the source code given in the `github`, but with the `level` attribute of the function `k3d.marching_cubes` changed to be 0.05 in order to get the yellow level of the graph to appear. In the 2D graphs, the yz-plane has the most variation depending on which x value the plane resides at. In 3D, we can see that this is because the yz plane contains two parabolic shapes (one in blue and one in yellow). When a different x value is chosen, the plane will contain different values since it may reside in either the yellow region, the blue region, or neither. Therefore, 2D graph (c) depicts the potential as seen through the positive yellow surface where the potential is at a maximum at the center (this correlates to the parabolic shape of the plate. 2D graph (d) depicts the potential as seen through the negative blue surface where the potential is at a maximum magnitude at the center but appears inverted compared to 2D graph (c) since the surface displays a negative potential.



(a) The 3D graph as seen through the yellow surface. This is comparable to the viewpoint seen in 2D graph (c)



(b) The 3D graph as seen through the blue surface. This is comparable to the viewpoint seen in 2D graph (d)

Therefore, 2D graphs (a) and (b) correlate to the xz and xy planes which appear identical to each other from every angle. This explains their identical 2D graphs.