SU2019 VP260 Honors Physics I

PROJECT

NUMERICAL SOLUTION OF THE LAPLACE'S EQUATION USING THE RELAXATION METHOD

November 18, 2019

Name:	Cao Hangrui	曹航瑞	ID:	518370910123
Name:	Li Ruiyu	李睿玉	ID:	518021910206
Name:	Ouyang Wenbin	欧阳文彬	ID:	518021910945
Name:	Zhang Boming	张泊明	ID:	518370910136

We state that each of us has contributed equally to this project.

Sign:_____

University of Michigan - Shanghai Jiao Tong University Joint Institute (UM-SJTU JI)

Content

1 Introduction

In previous study, we have learned how to calculate electric potential, electric field only when the charge distribution are known. The methods including Coulomb Law, Gauss Law and integration methods. We have also learned the relation between the electric potential and electric field.

However, the charge distribution is not known in many practical cases. Instead, the electric potential of the boundary is known. In electrostatics conditions, the surface of a conductor is an equipotential surface, but we can only obtain the values case when the distribution of the electric charge in highly symmetric situations. When the distribution of the electric charge is not uniform, we need to find new methods to perform calculation.

Given a region of space where there is no electric charge in its interior and enclosed by one and more conductors maintained at fixed potentials. Then, the problems turns into solving the Laplace equation $\nabla^2 V = 0$. This project aims to find the numerical values of the potential as a function of position in a certain region. We will use computer to help us perform the calculation numerically.

2 Methods

2.1 Related Theory

To solve this problem numerically, we should use following principle: In a region where there is no charge, the value of the electric potential at a given point is equal to the average value of the potential at surrounding points. The electric field components can be described using the partial derivative of the potential:

$$\mathbf{E} = -\nabla V \tag{1}$$

The Gauss theorem gives us following equation:

$$\nabla \cdot E = \frac{\rho}{\varepsilon_0} \tag{2}$$

We would consider the 2D case first. Namely, the electric potential will depend on x and y. For this case, we can enclose a point P (x,y,z) with a Gaussian surface in the shape of a cubical box shown below.

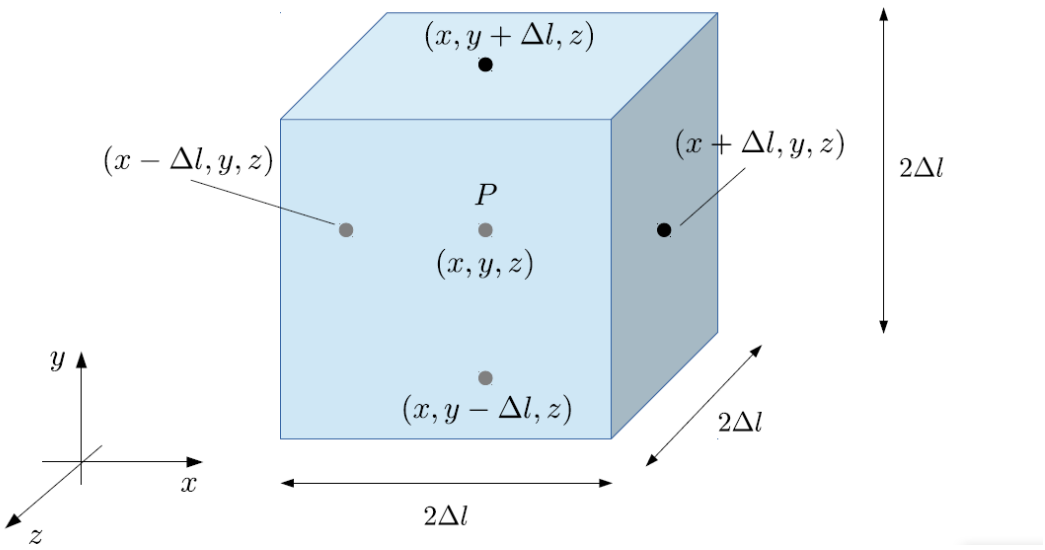

Figure 1: The Gaussian surface and the region[1]

For simplicity, we first consider the case there is no charge in the volume enclosed by this box. Since V is independent of z, from equation 1, the electric field components along z axis is zero. So there is no electric flux through xy plane.

The side $2\Delta L$ is chosen to be an infinitesimal value, so we can regard the flux through a surface approximately as $\mathbf{E} \cdot (2\Delta L)^2$. Then the total flux is:

$$\begin{aligned}\Phi_E =& E_x(x + \Delta l, y, z)(2\Delta l)^2 + (-E_x(x - \Delta l, y, z))\,(2\Delta l)^2 + \\ &+ E_y(x, y + \Delta l, z)(2\Delta l)^2 + (-E_y(x, y - \Delta l, z))\,(2\Delta l)^2\end{aligned} \tag{3}$$

The Φ_E is zero in this case, since there is no charge enclosed. Then, using equation 1, we rewrite the E_x, and E_y:

$$\begin{aligned}E_x(x + \Delta l, y, z) &\approx -\frac{V(x + \Delta l, y) - V(x, y)}{\Delta l} \\ E_x(x - \Delta l, y, z) &\approx -\frac{V(x, y) - V(x - \Delta l, y)}{\Delta l} \\ E_y(x, y + \Delta l, z) &\approx -\frac{V(x, y + \Delta l) - V(x, y)}{\Delta l} \\ E_y(x, y - \Delta l, z) &\approx -\frac{V(x, y) - V(x, y - \Delta l)}{\Delta l}\end{aligned} \tag{4}$$

Through substitution, we can obtain:

$$\begin{aligned} -[V(x+\Delta l, y) - V(x, y)] + [V(x, y) - V(x-\Delta l, y)] + \\ -[V(x, y+\Delta l) - V(x, y)] + [V(x, y) - V(x, y-\Delta l)] = 0 \end{aligned} \tag{5}$$

Then we obtain the electric potential at point P:

$$V(x, y) = \frac{1}{4}[V(x+\Delta l, y) + V(x-\Delta l, y) + V(x, y+\Delta l) + V(x, y-\Delta l)] \tag{6}$$

From equation 6, we can see the electric potential at a given point is the average value of the neighborhood four points.

2.2 Algorithms

To find the potential V of the interior volume of one given 2D box, where the boundary value of V is know, we can apply following approximation methods.

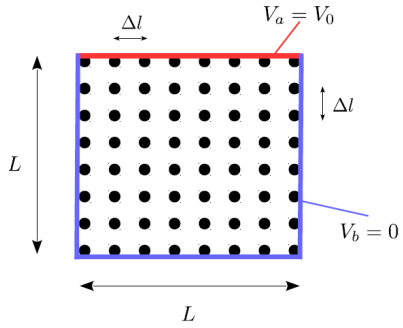

Figure 2: Grid points in a given box

The steps are as follows:

1. Choose a positive value of potential difference V_0.
2. Choose m^2 grid points inside the region. Then there are $(m-2)^2$ total points inside the box.
3. Label each point by pair (j, k), where $1 \leq j \leq m$ and $1 \leq k \leq m$. j refers to the jth column, k refers to the kth row of the grid.
4. For each grid, assign the initial value for it. First, set the initial values for all boundary points. Then for other values, the initial values can be chosen randomly. However, if the closer the chosen initial value is to the actual one, the less iteration steps will be taken. Here, we first set the initial value as $\frac{V_0}{2}$. It should be noticed that we cannot

set the initial potential to be zero, otherwise it will cause trouble in the iteration steps.

5. For a $V(j,k)$, the new value at that point is given by following formula:

$$V_{\text{new}}(j,k) = \frac{1}{4}[V(j+1,k) + V(j-1,k) + V(j,k+1) + V(j,k-1)] \tag{7}$$

6. We repeat the step 6, until for all the points, the value

$$\left| \frac{V_{\text{new}}\,(j,k) - V(j,k)}{V(j,k)} \right|$$

is less than the chosen accuracy ε.

For other shapes of boundaries, they can be rectangles or polygons, as long as we set the initial value well. Once we have determined the potential at all points, the electric field can be obtained as follows:

$$\begin{aligned} E_x(x,y) &= -\left.\frac{\partial V}{\partial x}\right|_{(x,y)} \approx -\frac{V(x+\Delta l, y) - V(x-\Delta l, y)}{2\Delta l} \\ E_y(x,y) &= -\left.\frac{\partial V}{\partial y}\right|_{(x,y)} \approx -\frac{V(x, y+\Delta l) - V(x, y-\Delta l)}{2\Delta l} \end{aligned} \tag{8}$$

3 Apply The Relaxation Method to Three Systems

3.1 System A & B

The setup for the two systems are shown in Figure 3 and 4. For system A, the top of the box is at V_a=1V and the other walls at V_b=0, while for system B, the top of the box is at V_a=0 and the other walls at V_b=1V.

3.1.1 The Electric Potential and Electric Field of Two Grid Density

To apply the relaxation method to solve the potential for the systems, we set the initial values of the potential inside the region all as 0.5V, with different initial V_a and V_b. Choose ε to be 10^{-7}, we used C++ to calculate electric potential inside the region for two different grid density: 50×50(m=50) and 200 × 200(m=200). We plotted the final results by visualizing the electric potential and equipotential electric filed in the form of a density plot and contour plot by Mathematica as shown below.

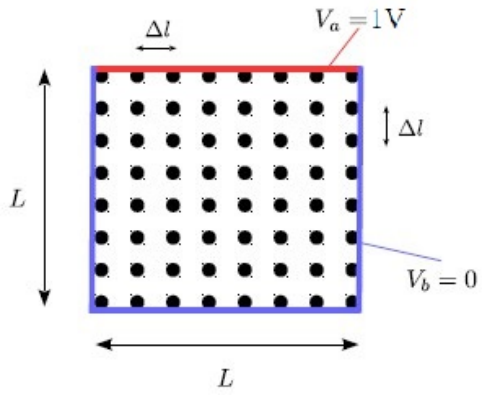

Figure 3: System A

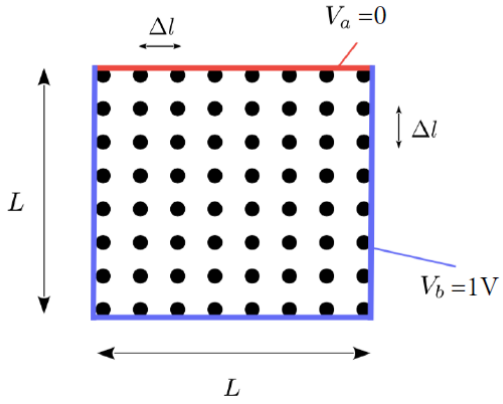

Figure 4: System B

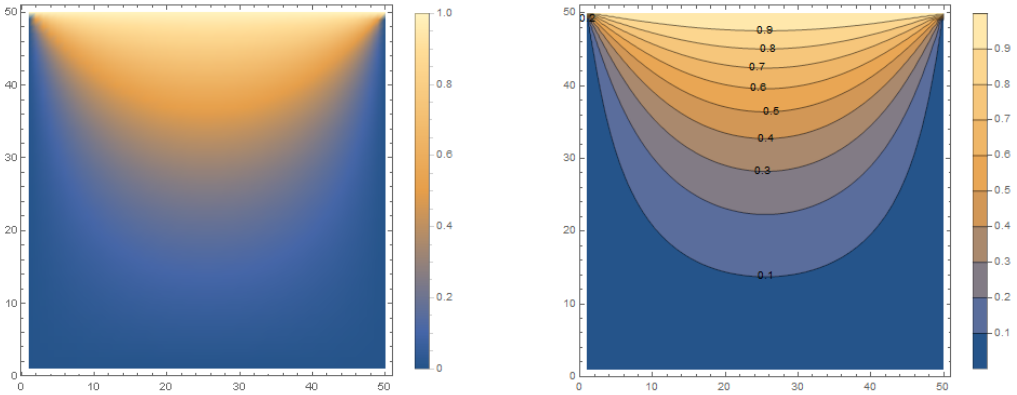

Figure 5: System A, $m = 50$, $\varepsilon = 10^{-7}$

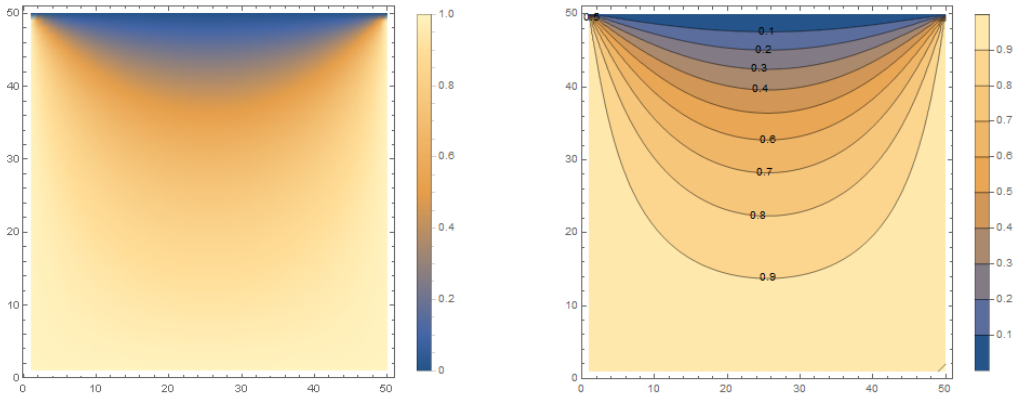

Figure 6: System B, $m = 50$, $\varepsilon = 10^{-7}$

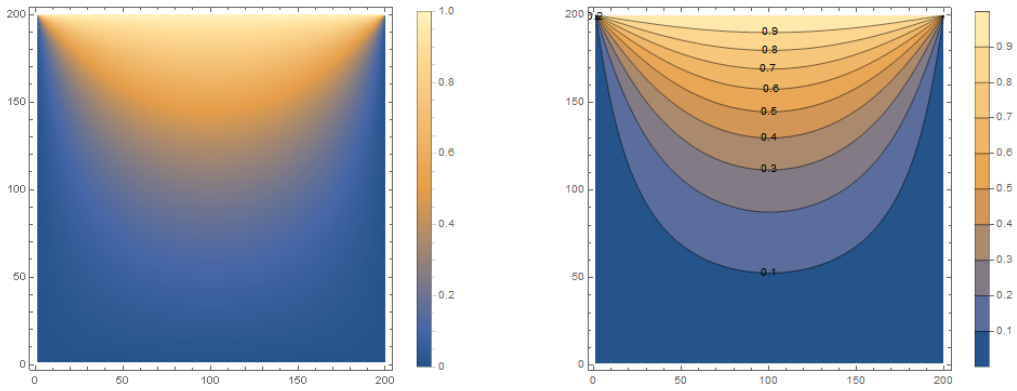

Figure 7: System A, $m = 200$, $\varepsilon = 10^{-7}$

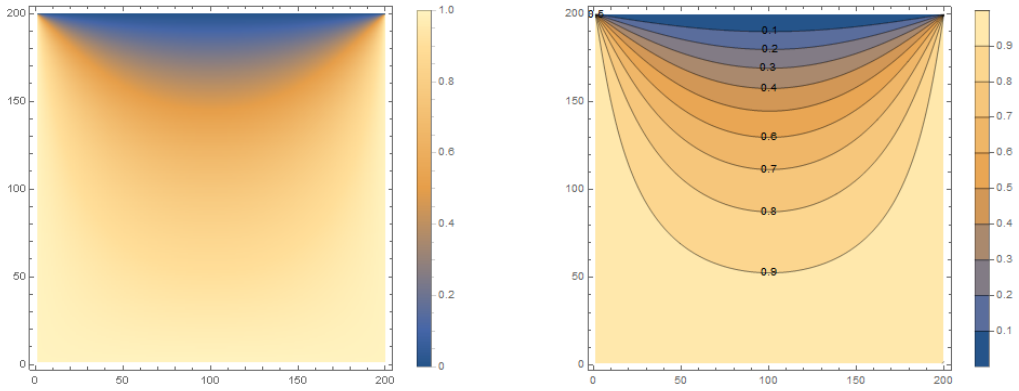

Figure 8: System B, $m = 200$, $\varepsilon = 10^{-7}$

From figures above, we found that the electric potential increases from the top side to the other three sides, shaped like a "V". Moreover, judging from the density of the equipotential lines, the rate at which the potential increases decreases from top to bottom and from the middle to the right and left sides, also shaped like a "V". Since the equipotential lines are denser near the top sides and are sparser in the interior, electric field is larger in magnitude around the top side, especially in the top-right and top-left concerns. Comparing the 4 figures with the results for potential of system A, we noticed that the general shape of equipotential lines are similar but the color which indicates magnitude of the potential is reverse.

Actually, potential of system B is the reverse of system A. The magnitude of potential at the same point in the system A and system B sums up to V_0=1V. This is reasonable because the configurations of system A and B are just reverse (structure of the system unchanged but values of V_a and V_b interchange).

According to equation 6, we calculated the corresponding electric field of system A & B and visualized it as Figure 9 to 12 has shown. From the figures, we found that electric filed is larger in magnitude near the top side, especially around the top-right and top-left corners and become smaller in the other three sides. The electric field always has a component pointing to the top side and a component pointing to the middle of the box, which corresponds to the potential change of the system as we discussed above. Comparing system A and B, the electric field at every point of A and B are of the same magnitude but opposite direction. This is also reasonable due to the similar structure but reserve boundary conditions of the two systems.

What's more, we notice that at the edge of the vector field, the vector is very large due to the initial value. Thus we ignore these values and make our figure more reasonable.

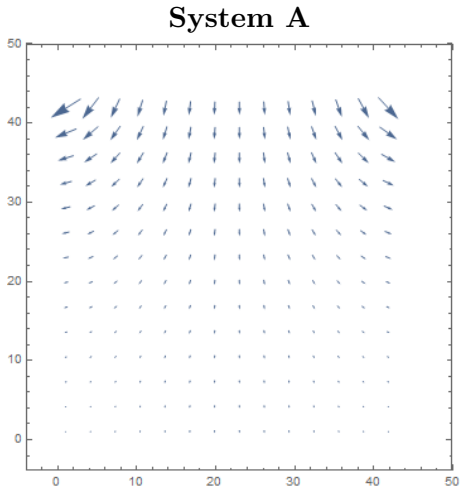

Figure 9: Electric field, $m = 50$, $\varepsilon = 10^{-7}$

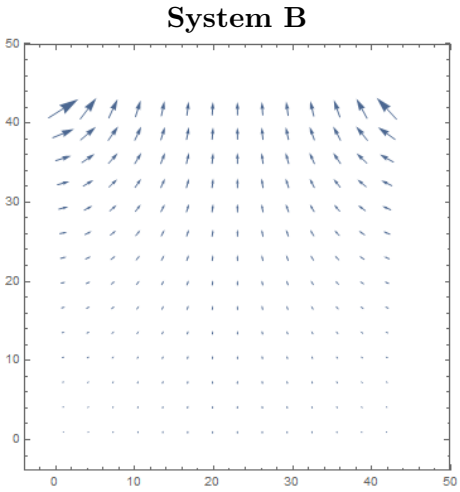

Figure 11: Electric field, $m = 50$, $\varepsilon = 10^{-7}$

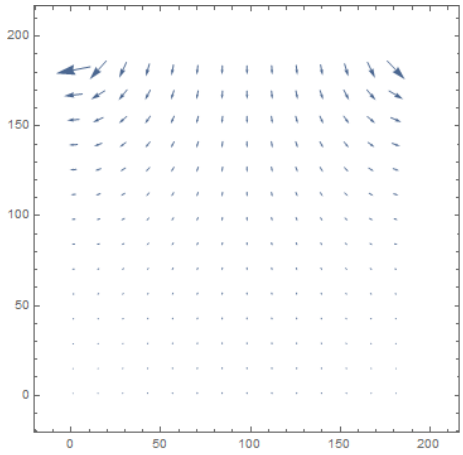

Figure 10: Electric field, $m = 200$, $\varepsilon = 10^{-7}$

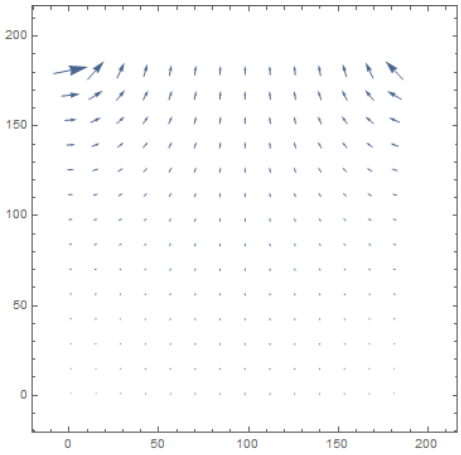

Figure 12: Electric field, $m = 200$, $\varepsilon = 10^{-7}$

3.1.2 Comparison for Different Values of ε

Choosing the denser grid size (200×200), we used the relaxation method to calculate the electric potential with three different accuracies: 10^{-1}, 10^{-3}, and 10^{-5}. The results are shown in the form of contour plots below.

System A

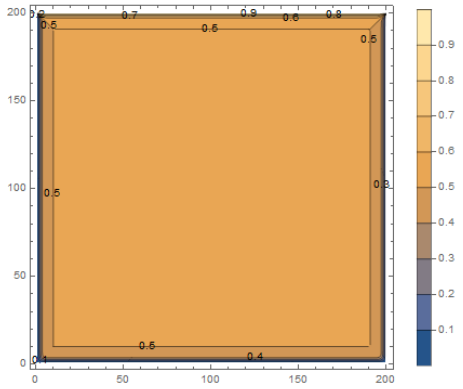

Figure 13: Contour plot, $m = 200$, $\varepsilon = 10^{-1}$

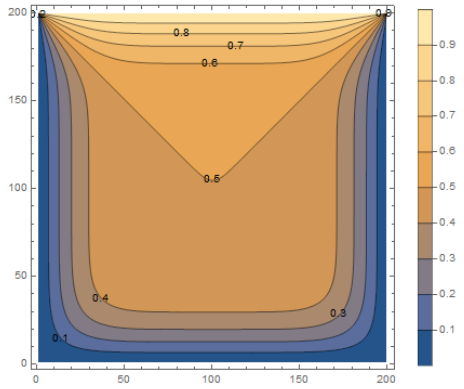

Figure 14: Contour plot, $m = 200$, $\varepsilon = 10^{-3}$

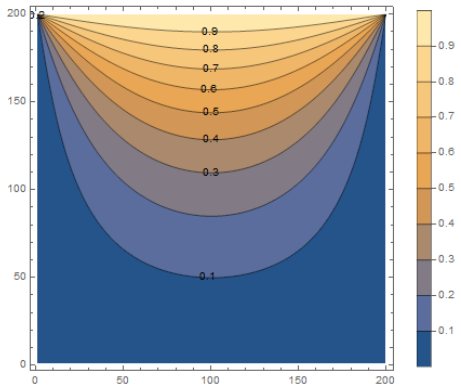

Figure 15: Contour plot, $m = 200$, $\varepsilon = 10^{-5}$

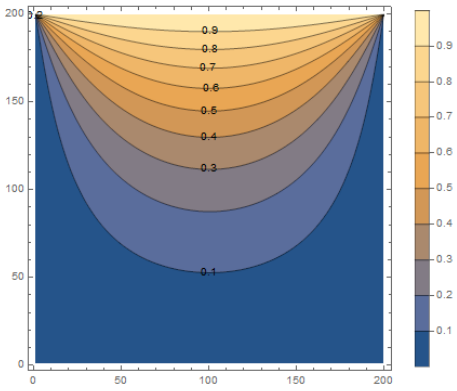

Figure 16: Contour plot, $m = 200$, $\varepsilon = 10^{-7}$

System B

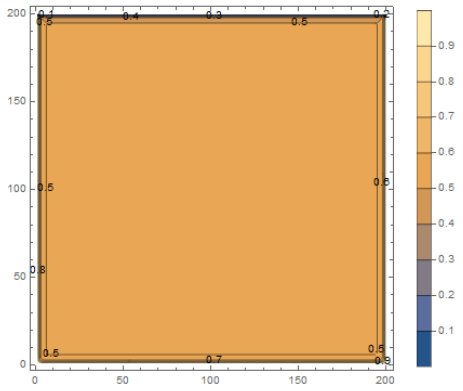

Figure 17: Contour plot, $m = 200$, $\varepsilon = 10^{-1}$

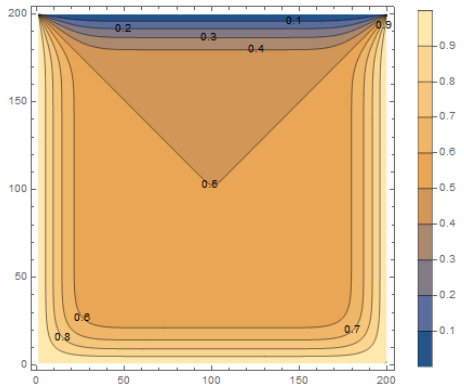

Figure 18: Contour plot, $m = 200$, $\varepsilon = 10^{-3}$

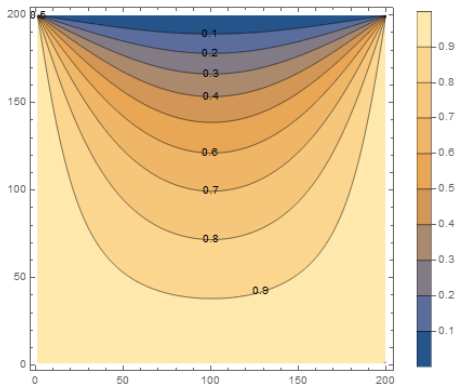

Figure 19: Contour plot, $m = 200$, $\varepsilon = 10^{-5}$

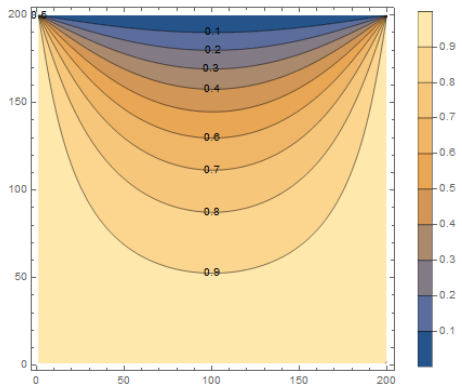

Figure 20: Contour plot, $m = 200$, $\varepsilon = 10^{-7}$

From the figures, we found that the smaller the accuracy ε is, the more accurate the calculated potential and the results are closer to the actual potential values. And for larger ε, the result will be affected more by the initial value, which is quiet reasonable according to the algorithm we use. What's more, the results for system A & B still shows good property of reverse.

3.1.3 Table of results and Discussion

The performance of the algorithm with two different grid sizes (50×50 and 200 × 200) and 4 different accuracies (10^{-1}, 10^{-3}, 10^{-5} and 10^{-7}) is tested. The results are listed in Table 1.

		System A		System B	
m	ε	# Iterations	Time[s]	# Iterations	Time[s]
50	10^{-1}	8	0.000	4	0.000
50	10^{-3}	846	0.004	275	0.001
50	10^{-5}	3221	0.013	2367	0.009
50	10^{-7}	5462	0.021	4605	0.021
200	10^{-1}	8	0.002	4	0.001
200	10^{-3}	1000	0.184	499	0.074
200	10^{-5}	30153	4.690	16127	2.070
200	10^{-7}	67653	9.299	53519	7.122

Table 1: Algorithm performance under different grid sizes and accuracies

We have the following observations form the table:

- The bigger the grid size m is, the larger numbers of iterations and the longer time needed to complete the calculations.
- The smaller the accuracy ε is, the larger numbers of iterations and the longer time needed to complete the calculations.
- Keep m unchanged and decrease ε, the numbers of iterations increase relatively slowly as approximately $\log(1/\varepsilon)$.
- Keep ε unchanged and increase m, the numbers of iterations increase relatively fast as approximately m^2.
- The iteration times for system B is always smaller than system A. Genreally speaking, the larger ε is, the larger ratio of difference is.

Therefore, the value of ε should not be too large and the value of m should not be too small, otherwise it will takes a lots of time for the algorithm to perform the calculations. But if the ε is too large or m is too small, the results would be fairly rough, which can not represent the real case. Thus, when applying this algorithm, we should choose both ε and m to be in a moderate range.

3.1.4 Illustrations of the Algorithm

To illustrate progress of the relaxation method for System A & B, we choose the dense grid size (200×200) and the accuracy $\varepsilon = 10^{-5}$ and plot the corresponding density plots in system B with equipotential lines for the electric potential at four different stages: the initial stage of the progress of the algorithm, the stage at one third of the iteration progress, the stage at two thirds of the iteration progress and the final stage (Figures 3.1.4, 3.1.4,

3.1.4 and 3.1.4). The reason why we choose $\varepsilon = 10^{-5}$ is that for $\varepsilon = 10^{-7}$, almost the latter half of the iteration process is finetuning.

System A

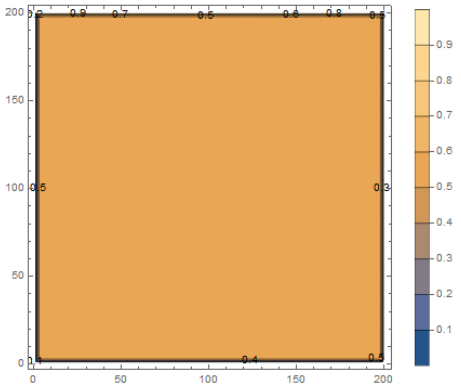

Figure 21: Density plot: initial stage

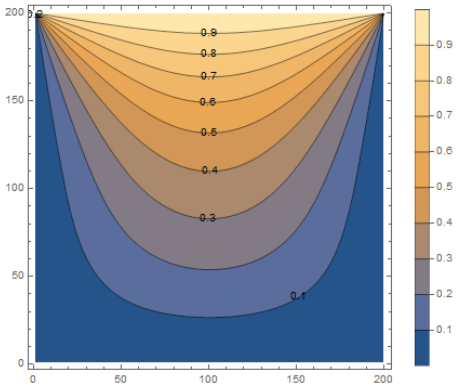

Figure 22: Density plot: transition 1

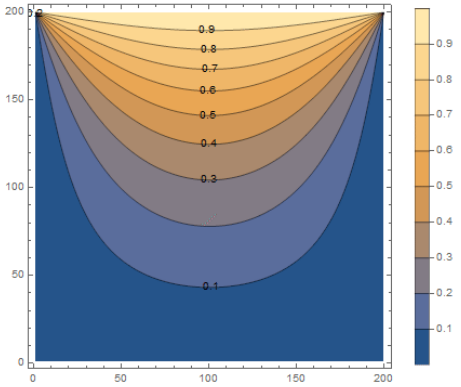

Figure 23: Density plot: transition 2

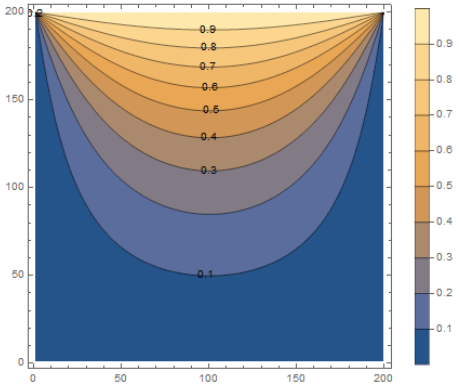

Figure 24: Density plot: final stage

System B

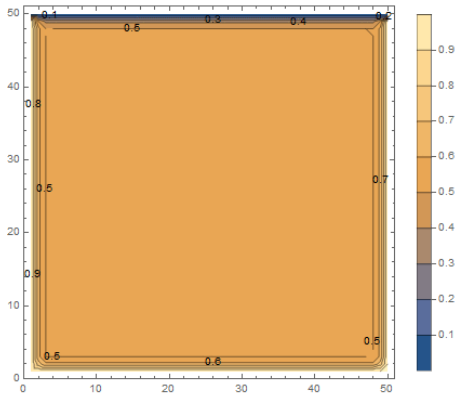

Figure 25: Density plot: initial stage

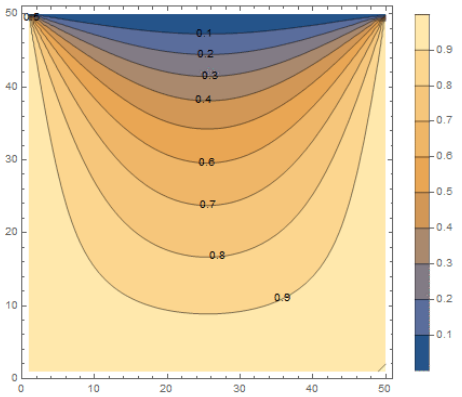

Figure 26: Density plot: transition 1

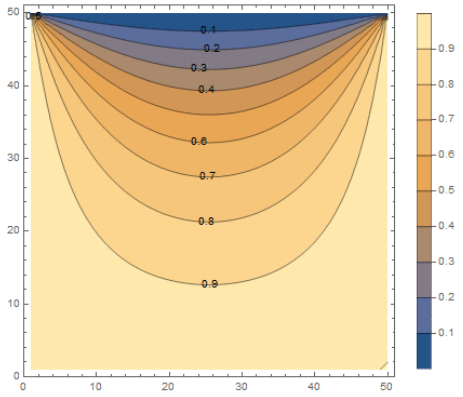

Figure 27: Density plot: transition 2

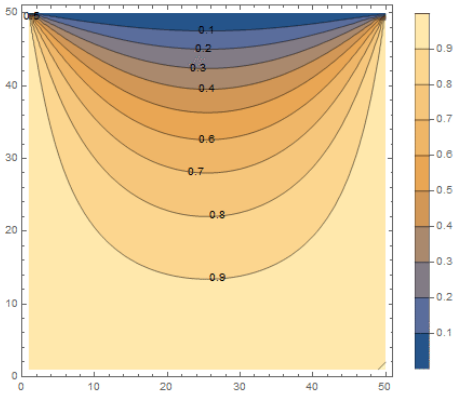

Figure 28: Density plot: final stage

The 4 figures shows that, the potential in the top side reduce the potential of the points around it, while the potential in the other three sides increase the potential of the points around them. As the iterations go on, the potential values at each point will gradually converge to the real values and the equipotential lines become smoother.

Animations of the progress is also provided in the submitted file for the project to better illustrate the progress of the iteration. (Named as with suffix ".gif".)

3.2 System C

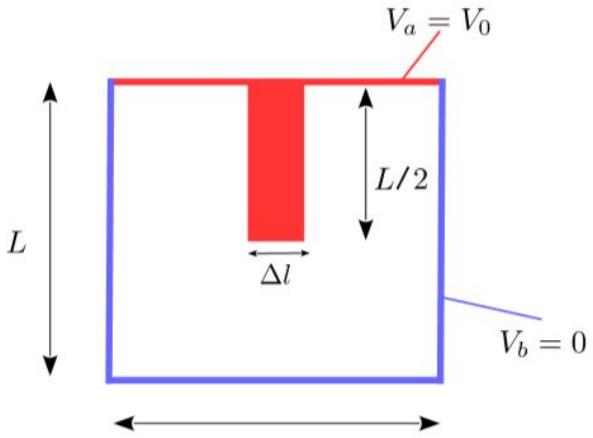

Figure 29: The initial setting of system C

The rough shape of system C is a box, the same as system A and system B. What has modified is that on the middle of the interior upper surface, there is a rectangular region with width Δl and length $L/2$ setted as $V = V_a = V_0$ (as shown in the Figure 29). And the potential of the walls on the left, right and bottom are all at 0.

Then in terms of applying the relaxation method, we choose the the sparse grid as a square of 50×50, and the dense grid as a square of 200×200. And the accuracy we used here is set as:

$$\varepsilon = 1e - 7$$

To simulate the rectangle area with a width Δl as shown in the figure 29, we set the potential of the $2 \times L/2$ region to be V_a (located on the top wall pointing down). Notice that we did not choose ΔL to be "1" but rather "2", which is due to maintaining the symmetry, since the square we choose here is a square of even number length edge. For other inner points, we set their initial potential as:

$$V = \frac{V_a + V_b}{2}$$

3.2.1 The Electric Potential and Electric Field of Two Grid Density

After we have set the initial condition for potential of the system C, we now run the program to apply the relaxation method. After a few minutes, our program come up to

the relative electric potential with the assumed $\varepsilon = 1e - 7$. Here, we present the electric potential by using density plot and contour plot:

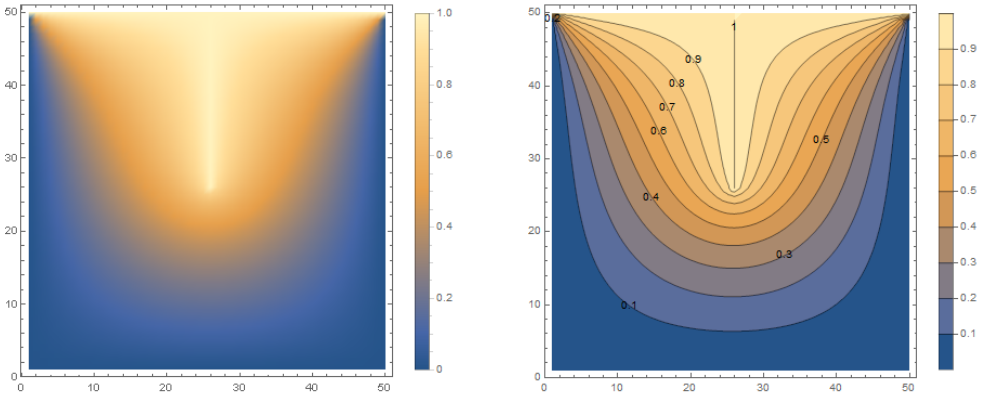

Figure 30: The density plot and contour plot for the sparse grid

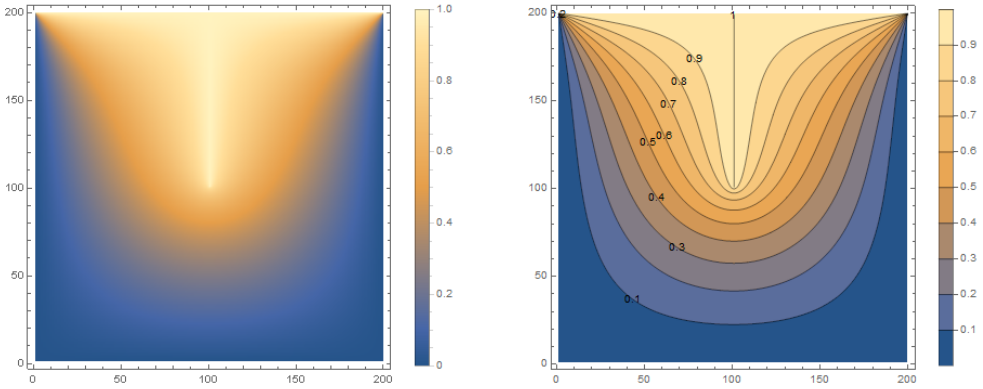

Figure 31: The density plot and contour plot for the dense grid

For the density figure, we obtain that the potential decreases as it goes downward or away from the center. Moreover, the potential is symmetric to the vertical axis in the middle. When comparing two figures between with the dense grid and the sparse grid, we notice that the potential turn more slowly and smoothly with the dense grid than the sparse grid.

For the contour plot, as expected, a somehow abrupt equipotential line occurs representing the $\Delta L \times L/2$ rectangle region pointing downward. Moreover, the equipotential lines are intensive on the upper two corners and near the bottom of the rectangle region. Also every

equipotential line is symmetric corresponding to the rectangle area, which is coincide with the symmetry of the density figure.

When comparing to the system A, we can see that in both systems, they're symmetric corresponding to the same axis. But the decreasing of the potential in the vertical axis is slower in system C, and the equipotential lines are bent more to avoid any intersection with the $\Delta L \times L/2$ rectangle region.

Then by applying equation xxx, we could obtain the corresponding figure of the electric vector field (with both direction and magnitude):

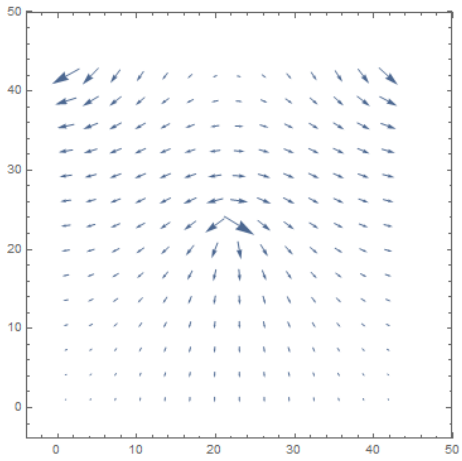

Figure 32: The electric vector field for sparse grid

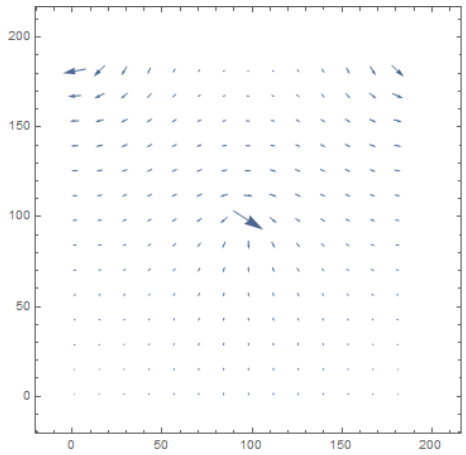

Figure 33: The electric vector field for dense grid

As we can see from 32 and 33, the vector is pointing basically downward and away from the center, which is coincide with the potential field as pointing from the higher potential to the lower potential. As for the difference between figure 32 and figure 33, it seems that the magnitude of vectors in the figure 32 is bigger than that in the figure 33. But in fact, the size of the figure is not the same since the difference between the size of sparse grid and the dense grid. So the vector arrows seem to be bigger in the figure 32, but in fact they're in the same level so magnitude. And the vector arrows turn their directions and vary their magnitude some smoothly in the dense grid situation.

For the difference between the system A, we notice that the electric vectors turn their and gradually point upward as the it gets near the center from the left or right side. But in the upper part of system C, due to the $\Delta L \times L/2$ rectangle region, the electric vectors are always horizontal in the center.

3.2.2 Comparison of Different Values of ε

In this part, we aim to analyse the difference of the choice of the different values of ε. Here, we fix the grid as the dense one, and test 3 more ε as ε_1 = 1e-1, ε_2 = 1e-3, ε_3 = 1e-5 for the potential field:

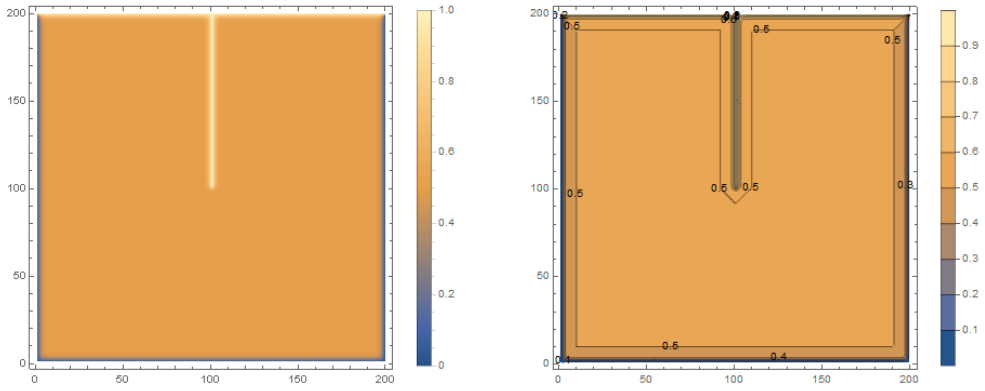

Figure 34: The density plot and contour plot for the ε_1 = 1e-1

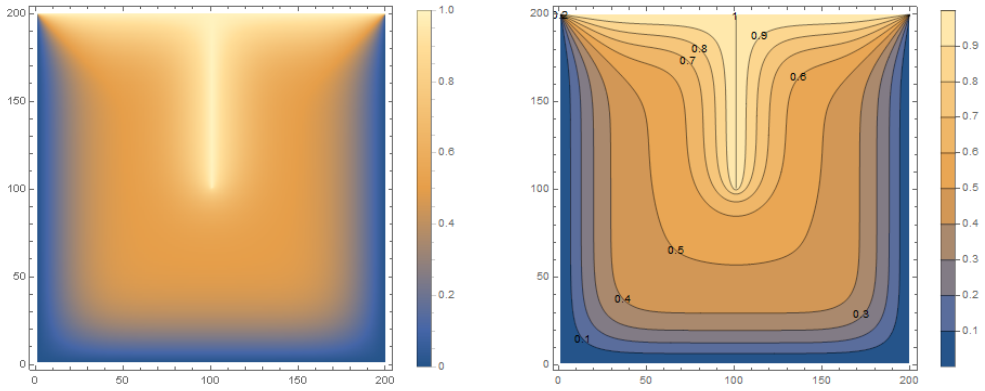

Figure 35: The density plot and contour plot for the ε_2 = 1e-3

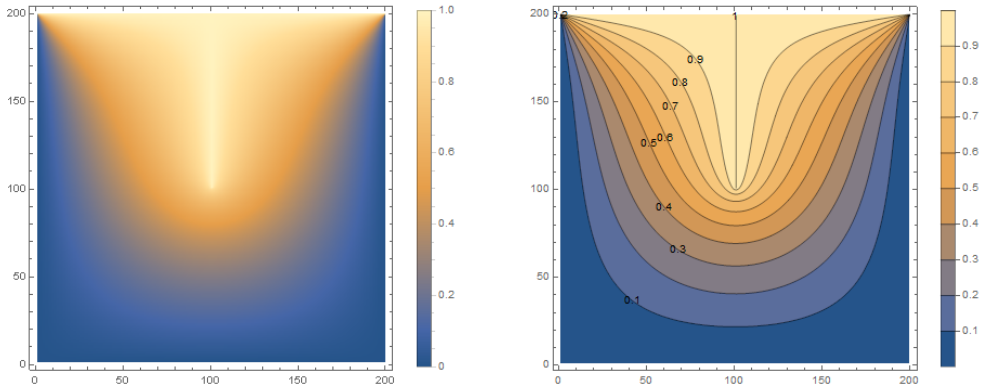

Figure 36: The density plot and contour plot for the ε_3 = 1e-5

As we can see, although the accuracy is difference, but the rough shapes are about the same: they're all symmetric about the vertical axis at the middle. But if we take a further inspection on the 3 groups of potential fields, we can find that the drops of the potential filed is relative fast in a big ε when comparing to a small one. This can also be noticed from the equipotential lines, as the lines are relatively dense in a big ε when comparing to a small one.

As for the ε_1 = 1e-1 and ε_2 = 1e-3, the differences from ε = 1e-7 are relatively large and obvious. But after the ε reaches 1e-5, the difference between it and ε = 1e-7 is relatively small and hard to be noticed.

When compared to the system A, the difference among the choices of ε is relative smaller. Namely, we could capture a big difference if we choose the ε to be 1e-1, 1e-3 and 1e-5, while the difference is relative small in the system C. Namely, in the system C, only a relative smaller ε needed to be chosen to get the same simulation degree when comparing to the system A.

3.2.3 Table of results and Discussion

In this part, we mainly focus on the analysis of the algorithm, namely, analysis its complexity and its relationship between the parameters such as the density of the gird and the accuracy. First, we demonstrate the information table, including the grid size m, accuracy ε, running time T, and the iteration time as:

m	ε	# Iterations	Time[s]
50	10^{-1}	8	0.000
50	10^{-3}	572	0.007
50	10^{-5}	1920	0.014
50	10^{-7}	3250	0.018
200	10^{-1}	8	0.002
200	10^{-3}	998	0.130
200	10^{-5}	17749	2.282
200	10^{-7}	39655	5.477

Table 2: Algorithm performance under different grid sizes and accuracies

First, for the running time, qualitatively, it's relatively short when the grid size is 50, and it increase to a few seconds when the size is 200, which is still doable and effective. But the increasing speed is fast, and maybe when the size is reaching for about 1000, it will take too much time to realize the algorithm.And for the accuracy, when ε getting small, there's some but not significant increasing of the running time.

Second, for the accuracy, namely we fix the grid size:

$$log(\frac{1}{1e-3}) - log(\frac{1}{1e-1}) = 3 - 1 = 2$$

$$I_{sparse}\Big|_{\varepsilon=1e-3} - I_{sparse}\Big|_{\varepsilon=1e-1} = 572 - 8 = 564$$

$$log(\frac{1}{1e-5}) - log(\frac{1}{1e-3}) = 5 - 3 = 2$$

$$I_{sparse}\Big|_{\varepsilon=1e-5} - I_{sparse}\Big|_{\varepsilon=1e-5} = 1920 - 572 = 1348$$

$$log(\frac{1}{1e-7}) - log(\frac{1}{1e-5}) = 7 - 5 = 2$$

$$I_{sparse}\Big|_{\varepsilon=1e-7} - I_{sparse}\Big|_{\varepsilon=1e-5} = 3250 - 1920 = 1330$$

If we neglect the situation when $\varepsilon = 1e-1$, then we could find out that when the accuracy decrease to its 0.01 times, then the the increasing of the iteration time is about the same. And further this would points to a guess: whether the iteration time I has a relation with the grid size and the accuracy as:

$$I = O\left(m^2 \times log\left(\frac{1}{\varepsilon}\right)\right) \tag{9}$$

This will be roughly discussed in the discussion section.

3.2.4 Illustration of the Algorithm

As to better demonstrate the process of the algorithm, we obtain the relative one plot right after the beginning, two plots during the process and one at the very end:

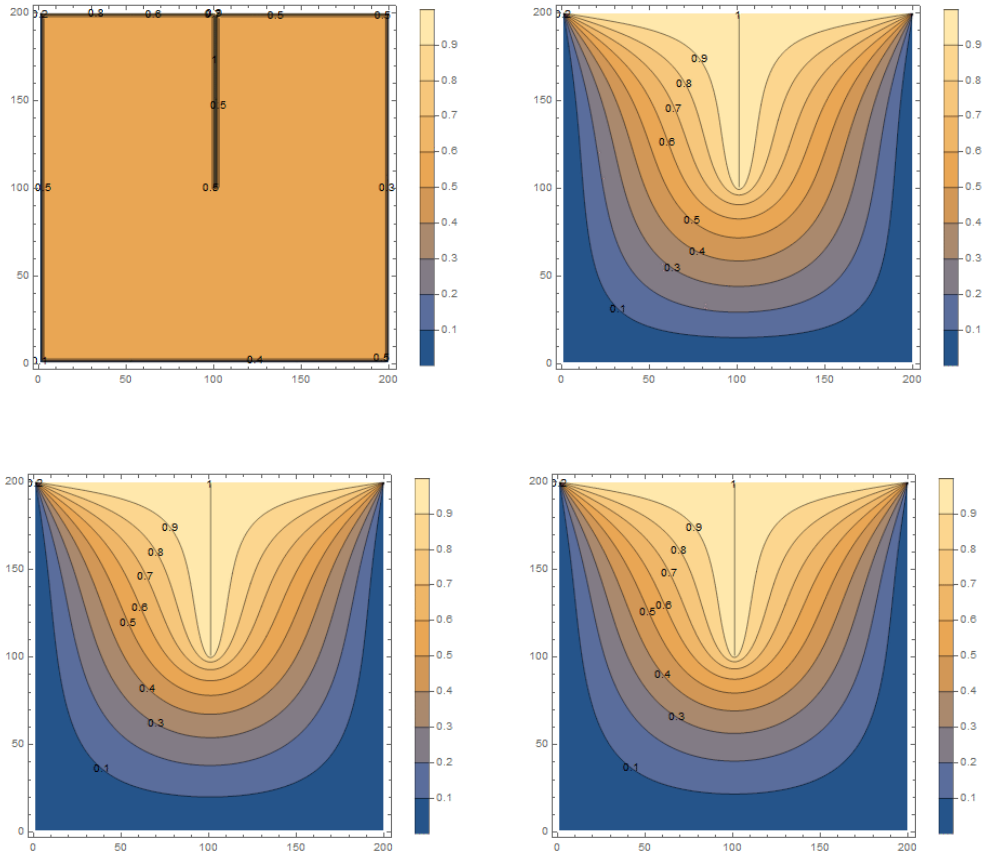

Figure 37: The four stages of the process of the algorithm

Roman was not built in one day, and the potential field is not obtained out of the air. Instead, we can see that the potential field converges gradually, and in this case, symmetrically.

Also we have made fancy gif figures to show the process of the algorithm, and they're at the attached file.

4 Discussions

4.1 Exploration of the complexity of the relaxation method

In this system C, we have a guess on the complexity of the relaxation method on the system C, as equation 9. But in fact, the shape of the system C is not so general, and we now turn into system A, a more straight forward square shape. Here we want have a rough experimentally verification of the equation:

$$I = O\left(m^2 \times log\left(\frac{1}{\varepsilon}\right)\right)$$

where the I is the iteration time and the m is the size of the grid, and ε is the accuracy. First, we fix accuracy as 1e-5 and try to explore the relation between m and I. We make the m vary from 51 to 200, and simulate the algorithm and calculate the corresponding iteration time I, as shown in the table 3.

m	I	m	I	m	I	m	I	m	I	m	I
51	3334	76	6572	101	10504	126	14967	151	19838	176	25003
52	3447	77	6718	102	10672	127	15156	152	20039	177	25215
53	3563	78	6863	103	10843	128	15344	153	20241	178	25426
54	3679	79	7011	104	11013	129	15533	154	20443	179	25639
55	3798	80	7159	105	11186	130	15722	155	20647	180	25851
56	3917	81	7309	106	11358	131	15913	156	20850	181	26064
57	4038	82	7459	107	11532	132	16103	157	21054	182	26276
58	4159	83	7611	108	11705	133	16295	158	21258	183	26490
59	4283	84	7763	109	11881	134	16487	159	21463	184	26703
60	4408	85	7917	110	12056	135	16680	160	21668	185	26918
61	4534	86	8071	111	12233	136	16873	161	21874	186	27131
62	4661	87	8227	112	12410	137	17067	162	22080	187	27346
63	4790	88	8383	113	12588	138	17261	163	22286	188	27560
64	4920	89	8541	114	12767	139	17457	164	22493	189	27775
65	5051	90	8699	115	12947	140	17652	165	22700	190	27990
66	5183	91	8859	116	13126	141	17848	166	22908	191	28206
67	5317	92	9019	117	13308	142	18044	167	23116	192	28421
68	5452	93	9181	118	13489	143	18242	168	23324	193	28637
69	5588	94	9343	119	13672	144	18439	169	23533	194	28852
70	5725	95	9506	120	13854	145	18638	170	23742	195	29069
71	5864	96	9670	121	14039	146	18836	171	23952	196	29285
72	6003	97	9835	122	14223	147	19036	172	24161	197	29502
73	6144	98	10000	123	14408	148	19235	173	24371	198	29719
74	6285	99	10168	124	14594	149	19436	174	24581	199	29936
75	6429	100	10335	125	14781	150	19636	175	24793	200	30153

Table 3: m vs. I

Then we apply a linear fit to the m and I, and obtained:

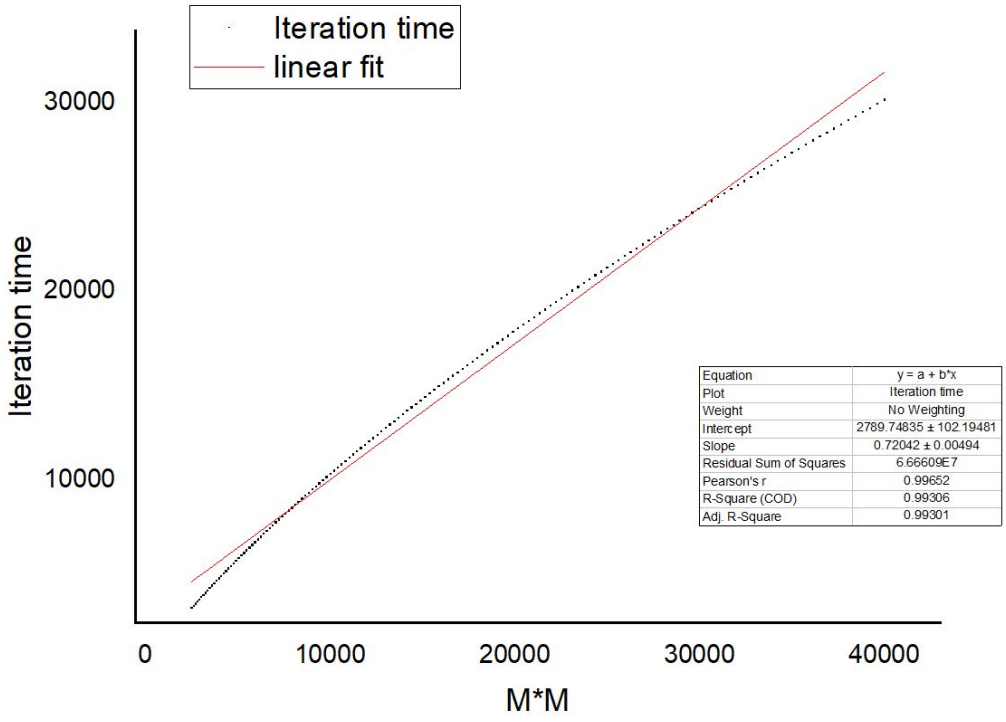

Figure 38: Linear fit to m*m and I

As we can see in the figure 38, the R-Square is 0.993 very close to 1, and the slop is about 0.7, relative small comparing to the value m. Therefore, it shows evidence, to some extent, to equation 9.

Second we fix m as 80 and try to explore the relation between ε and I. We make the $\log(1/\varepsilon)$ vary from 3 to 10.4, and simulate the algorithm and calculate the corresponding iteration time I, as shown in the table 4.

ε	$\log(1/\varepsilon)$	I	ε	$\log(1/\varepsilon)$	I	ε	$\log(1/\varepsilon)$	I
0.001	3	996	3.16E-06	5.5	8625	1E-08	8	15908
0.000794	3.1	1240	2.51E-06	5.6	8917	7.94E-09	8.1	16199
0.000631	3.2	1520	2E-06	5.7	9209	6.31E-09	8.2	16490
0.000501	3.3	1828	1.58E-06	5.8	9501	5.01E-09	8.3	16781
0.000398	3.4	2152	1.26E-06	5.9	9792	3.98E-09	8.4	17073
0.000316	3.5	2486	0.000001	6	10084	3.16E-09	8.5	17364
0.000251	3.6	2824	7.94E-07	6.1	10375	2.51E-09	8.6	17655
0.0002	3.7	3158	6.31E-07	6.2	10667	2E-09	8.7	17946
0.000158	3.8	3487	5.01E-07	6.3	10958	1.58E-09	8.8	18237
0.000126	3.9	3812	3.98E-07	6.4	11249	1.26E-09	8.9	18528
0.0001	4	4132	3.16E-07	6.5	11541	1E-09	9	18819
7.94E-05	4.1	4447	2.51E-07	6.6	11832	7.94E-10	9.1	19110
6.31E-05	4.2	4759	2E-07	6.7	12123	6.31E-10	9.2	19402
5.01E-05	4.3	5066	1.58E-07	6.8	12414	5.01E-10	9.3	19693
3.98E-05	4.4	5371	1.26E-07	6.9	12705	3.98E-10	9.4	19984
3.16E-05	4.5	5673	1E-07	7	12997	3.16E-10	9.5	20275
2.51E-05	4.6	5973	7.94E-08	7.1	13288	2.51E-10	9.6	20566
2E-05	4.7	6272	6.31E-08	7.2	13579	2E-10	9.7	20857
1.58E-05	4.8	6568	5.01E-08	7.3	13870	1.58E-10	9.8	21148
1.26E-05	4.9	6864	3.98E-08	7.4	14161	1.26E-10	9.9	21440
0.00001	5	7159	3.16E-08	7.5	14452	1E-10	10	21731
7.94E-06	5.1	7453	2.51E-08	7.6	14744	7.94E-11	10.1	22022
6.31E-06	5.2	7747	2E-08	7.7	15035	6.31E-11	10.2	22313
5.01E-06	5.3	8040	1.58E-08	7.8	15326	5.01E-11	10.3	22604
3.16E-11	10.5	23186	2.51E-11	10.6	23477	2E-11	10.7	23769
3.98E-06	5.4	8333	1.26E-08	7.9	15617	3.98E-11	10.4	22895

Table 4: $\log(1/\varepsilon)$ vs. I

Then we apply a linear fit to the $\log(1/\varepsilon)$ and I, and obtained:

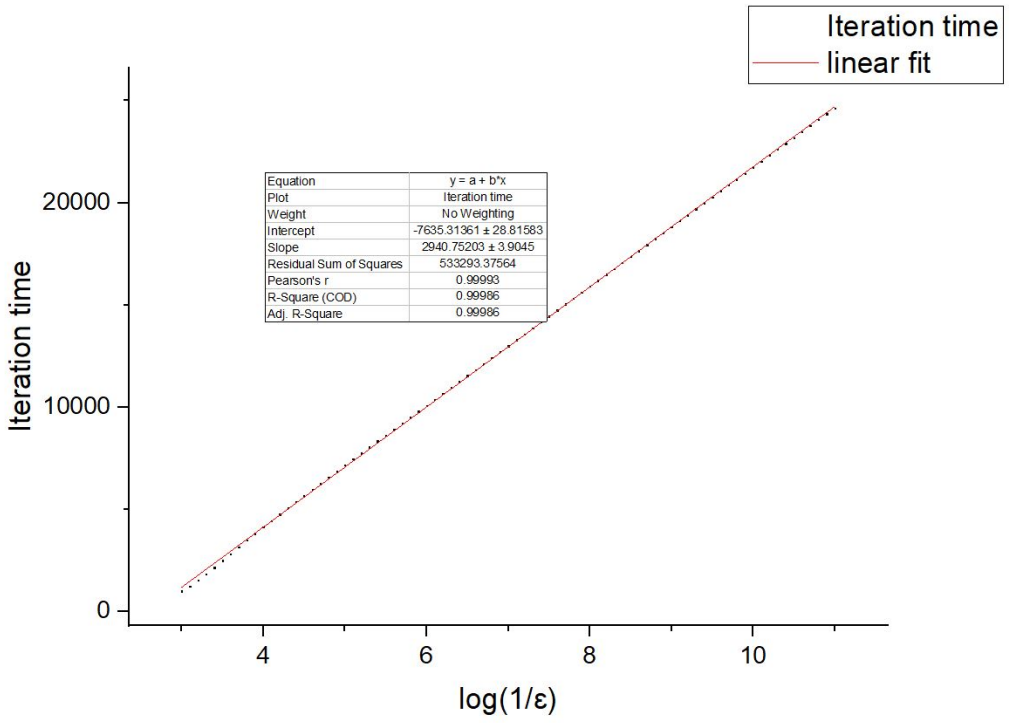

Figure 39: Linear fit to $\log(1/\varepsilon)$ and I

As we can see in the figure 39, the R-Square is 0.999 very close to 1. It shows that I has a linear relation with $\log(1/\varepsilon)$. Therefore, it shows evidence, to some extent, to equation 9.

4.2 Generalization to Poission's Equation

We have studied the electric field inside the conductor, where the electric charge density should be zero. Here we explore the distribution for general cases, where the electric charge density is not zero. Generally, we have:

$$\nabla^2 V = -\frac{\rho}{\varepsilon_0} \tag{10}$$

In this case, the total flux of the small cubic box of side $2\Delta L$ will not be zero.

$$\begin{aligned}\Phi_E =& E_x(x+\Delta l, y, z)(2\Delta l)^2 + \left(-E_x(x-\Delta l, y, z)\right)(2\Delta l)^2 + \\ &+ E_y(x, y+\Delta l, z)(2\Delta l)^2 + \left(-E_y(x, y-\Delta l, z)\right)(2\Delta l)^2 \\ &= \frac{q_{enclosed}}{\varepsilon_0} = \frac{\rho(x,y,z)\times 8\Delta l^3}{\varepsilon_0}\end{aligned}$$

Write the electric field components using approximation formula(like the one in method section), we can deduce the interation formula:

$$V_{\text{new}}(j,k) = \frac{1}{4}[V(j+1,k) + V(j-1,k) + V(j,k+1) + V(j,k-1) - 2\Delta l^2 \frac{\rho_{i,j}}{\varepsilon_0}]$$

where Δl is the spacing constant we choose, equal to $1/(n+1)$. So this generalization provides us formula for the case where ρ is not zero.

4.3 Generalization to 3D case

We would derive the equation for 3D case in this part.

First, we consider the case of 1D case. In this case, the potential V will be a function of only one dimension. The Laplace equation, the laplace equation will be contracted to:

$$\frac{\partial^2 U}{\partial x^2} = 0 \tag{11}$$

To solve this equations, many methods can be applied. Since we have already used the box with side $2\Delta L$, we can set E_y and E_z both be zero. Then,$V(x)$ turns into:

$$V(x) = \frac{1}{2}(V(x+\Delta L) - V(x-\Delta L))$$

For for $V(i)$, we have:

$$V(i) = \frac{1}{2}(V(i+1) + V(i-1))$$

In 3D case, the potential V is a function of x, y and z. Then for the cubical box with side $2\Delta L$, the flux to the Gaussian surface that are parallel to xy-plane exist. The total flux is:

$$\begin{aligned}\Phi_E = {} & E_x(x+\Delta l, y, z)(2\Delta l)^2 + (-E_x(x-\Delta l, y, z))\,(2\Delta l)^2 \\ & + E_y(x, y+\Delta l, z)(2\Delta l)^2 + (-E_y(x, y-\Delta l, z))\,(2\Delta l)^2 \\ & E_z(x, y, z+\Delta l)(2\Delta l)^2 + (-E_z(x, y, z-\Delta l))\,(2\Delta l)^2 = 0\end{aligned} \tag{12}$$

In this case, we express E_z in following ways:

$$\begin{aligned}E_z(x, y, z+\Delta l) &\approx -\frac{V(x, y, z+\Delta l) - V(x, y, z)}{\Delta l} \\ E_y(x, y, z-\Delta l) &\approx -\frac{V(x, y, z) - V(x, y, z-\Delta l)}{\Delta l}\end{aligned} \tag{13}$$

Perform substitution and then we would get:

$$\begin{aligned}V(x, y, z) = {} & \frac{1}{6}[V(x+\Delta l, y, z) + V(x-\Delta l, y, z) + V(x, y+\Delta l, z) \\ & + V(x, y-\Delta l, z) + V(x, y, z+\Delta l) + V(x, y, z-\Delta l)]\end{aligned} \tag{14}$$

If we want to apply this for a cube, we can choose m^3 grid points, and label each grid point as (i, j, k). where $1 \leq j \leq m$, $1 \leq j \leq m$ and $1 \leq k \leq m$. Then when we are doing the iteration, the formula becomes:

$$V_{new}(i,j,k) = \frac{1}{6}(V(i{+}1,j,k){+}V(i,j{+}1,k){+}V(i,j,k{+}1){+}V(i{-}1,j,k){+}V(i,j{-}1,k){+}V(i,j,k{-}1) \tag{15}$$

4.4 Alternative method- using Convolution of Matrices

We can use the convolution of matrices to replace the original iterating function [2]. We use the 2D case as example.

For the 2D case, the new value of potential V at (i, j) is shown by equation 7. We use A_0 to denote the original matrix of all grid points. The size of it depends on the grid points we choose, and the value a_{ij} represents the electric potential at that point.

Then taking the average of the neighbor four points can be shown as:

$$A_{n+1} = A_n * B \tag{16}$$

where B is the matrix

$$A = \begin{pmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{pmatrix} \tag{17}$$

In principle, through applying this convolution, we just find the average value of each point. However, for Matlab and Mathematica, there exists some built-in function, such as 'conv2', which can be used to calculate the convolution, which simplifies the calculation process.

5 Conclusion

In this project, we study the numerical method to calculate the approximate value of potential in a region with certain electric potential at its boundary. Using the relaxation method, we choose many grid points. We perform iteration by taking the average of the four potentials of the surrounding points. When the maximum value of the potential between two adjacent points are less than a given value ϵ, we stop the iteration and get the potential values of all points.

Applying this method, we examine three systems with certain boundary potential values. Using Mathematica and C++ code, we solve the systems and use surface plot function to visualize the potential field. For the accuracy ϵ, we choose different values and provide corresponding plots.

To show the stages of the iteration process, we plot four stages of one system. We also record more states and make it a gif figure.

For the discussion part, we use algorithms to estimate the time complexity. We generalize the case to Poisson's Equation and 3D case. We also use convolution of the matrix to simplify the original algorithms.

6 References

[1].Krzyzosiak, Mateusz. 'Calculating Potentials Due to Charged Conductors: Numerical Solution of the Laplace's Equation Using the Relaxation Method'. University of Michigan-Shanghai Jiao Tong University Joint Institute.

[2]. Song Ho Ann. 'Examples of 2D convolution'.http://www.songho.ca/dsp/convolution.

7 Appendix

7.1 Computer's Configuration

All the codes are run on the following configuration:

CPU: Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s)

Memory: 8GB DDR4 2400Hz, Single channel

7.2 Codes

`proj.cpp`

```
#include <dir.h>
#include <array>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <iostream>
#include <iterator>
#include <sstream>
#include <string>
#include <thread>
#include <vector>

using namespace std;

#define TEST_TIME
// #define GIF

int const M = 50;
int const INVL = 200;

void initGrid(double grid[M][M], int model) {
    fill(grid[0], grid[0] + M * M, -1);
    switch (model) {
    case 1:
        for (int i = 0; i != M; ++i)
        for (int j = 0; j != M; ++j)
            if (i == 0)
```

```
            grid[i][j] = 1;
            else if (j == 0 || i == M - 1 || j == M - 1)
            grid[i][j] = 0;
        break;
    case 2:
        for (int i = 0; i != M; ++i)
        for (int j = 0; j != M; ++j)
            if (i == 0)
            grid[i][j] = 0;
            else if (j == 0 || i == M - 1 || j == M - 1)
            grid[i][j] = 1;
        break;
    case 3:
        for (int i = 0; i != M; ++i)
        for (int j = 0; j != M; ++j)
            if (i == 0)
            grid[i][j] = 1;
            else if (j == 0 || i == M - 1 || j == M - 1)
            grid[i][j] = 0;
        for (int i = 0; i != M / 2; ++i) grid[i][M / 2] = 1;
        for (int i = 0; M % 2 && i != M / 2; ++i) grid[i][M / 2 +
            1] = 1;
        break;
    default:
        break;
    }
}

void printGrid(double grid[M][M], string name, ostream &os) {
    os << name << " = {";
    for (int i = M - 1; i != -1; i--) {
    os << "{";
    for (int j = 0; j != M - 1; ++j) os << grid[i][j] << ",";
    os << grid[i][M - 1] << "}";
    if (i != 0) os << ",";
    }
    os << "};" << endl;
}

void printCode(double grid[M][M], int count, string name, ostream
    &os) {
```

```
    printGrid(grid, name, os);
    os << "Export[\"" << name << "_density.png\", ListDensityPlot
       [" << name
        << ", PlotLegends -> Automatic]];" << endl;
    os << "Export[\"" << name << "_contour.png\", ListContourPlot
       [" << name
        << ", PlotLegends -> Automatic, ContourLabels -> All]];"
           << endl;
// gif begin
#ifdef GIF
    // os << "Export[\"" << name << "_density.gif\",{";
    // for (int i = 0; i != count - 1; ++i)
    //   os << "ListDensityPlot[" << name + to_string(i)
    //      << ", PlotLegends -> Automatic],";
    // os << "ListDensityPlot[" << name + to_string(count)
    //    << ", PlotLegends -> Automatic]";
    // os << "}]" << endl;
    os << "Export[\"" << name << "_contour.gif\",{";
    for (int i = 0; i != count + 1; ++i)
    os << "ListContourPlot[" << name + to_string(i)
       << ", PlotLegends -> Automatic, ContourLabels -> All],";
    for (int i = 0; i != 10; ++i)
    os << "ListContourPlot[" << name + to_string(count)
       << ", PlotLegends -> Automatic, ContourLabels -> All],";
    os << "ListContourPlot[" << name + to_string(count)
       << ", PlotLegends -> Automatic, ContourLabels -> All]";
    os << "}]" << endl;
#endif
    // vector plot begin
    os << name << "vector = {";
    for (int i = M - 4; i != 1; --i) {
    os << "{";
    for (int j = M - 4; j != 1; --j) {
        os << "{" << 0.5 * (grid[j][i + 1] - grid[j][i - 1]) << "
           ,"
           << 0.5 * (grid[j + 1][i] - grid[j - 1][i]) << "}";
        if (j != 2) os << ",";
    }
    os << "}";
    if (i != 2) os << ",";
    }
```

```
    os << "};" << endl;
    os << "Export[\"" << name << "_vector.png\", ListVectorPlot["
       << name
       << "vector]];" << endl;
}

int iter(double grid[M][M], double const initGrid[M][M], double
   eps,
         string name, ostream &os) {
    double bufferGrid[M][M] = {0};
    int count = 0;
    for (int i = 0; i != M; ++i)
    for (int j = 0; j != M; ++j)
        grid[i][j] = initGrid[i][j] < 0 ? 0.5 : initGrid[i][j];
    for (int canExit = 0; !canExit; ++count) {
    canExit = 1;
    copy(initGrid[0], initGrid[0] + M * M, bufferGrid[0]);
    for (auto i = 0; i < M; i++) {
        for (auto j = 0; j < M; j++) {
        if (initGrid[i][j] >= 0) continue;
        bufferGrid[i][j] = (grid[i + 1][j] + grid[i - 1][j] +
           grid[i][j + 1] +
                            grid[i][j - 1]) /
                            4.f;
        if (canExit)
            if (fabs((bufferGrid[i][j] - grid[i][j]) / grid[i][j
               ]) > eps)
            canExit = 0;
        }
    }
    copy(bufferGrid[0], bufferGrid[0] + M * M, grid[0]);
#ifndef TEST_TIME
#ifdef GIF
    if (count % INVL == 0) printGrid(grid, name + to_string(count
        / INVL), os);
#endif
#endif
    }
    return count;
}
```

```
// the real eps will be 1e-[eps]
void generateData(double eps, int model) {
    double grid[M][M] = {0};
    double reGrid[M][M] = {0};
    int iterTime;
    clock_t start, end;
    stringstream name;
    name << M << "00" << eps;
    mkdir((to_string(model)).c_str());
    ofstream fs(to_string(model) + "/" + name.str() + ".wls");
    fs.precision(10);
    fs << fixed;
    initGrid(grid, model);
    start = clock();
    iterTime = iter(reGrid, grid, pow(0.1, eps), "data00" + name.
        str(), fs);
    end = clock();
    cout << "data00" << name.str() << " eps:" << pow(0.1, eps)
        << " time:" << (double)(end - start) / CLOCKS_PER_SEC
        << "s iter time:" << iterTime << " " << endl;
#ifndef TEST_TIME
    printCode(reGrid, iterTime / INVL - (iterTime % INVL == 0),
            "data00" + name.str(), fs);
    system(
        ("cd " + to_string(model) + " & wolframscript -f " + name
            .str() + ".wls")
             .c_str());
#endif
}

int main() {
    // vector<thread> myVec;
    // // for (int model = 1; model != 4; ++model)
    // int model = 3;
    // for (int eps = 1; eps != 9; ++++eps)
    //   myVec.push_back(thread(generateData, eps, model));
    // for (auto &thr : myVec) thr.join();
    // int eps = 5, model = 3;
    for (int model = 1; model != 4; ++model)
    for (int eps = 1; eps != 9; ++++eps) generateData(eps, model)
        ;
```

```
}
```

hh.cpp

```
#include <cmath>
#include <iostream>

using namespace std;

double a[3][1010][1010];
int M = 200;
double eps;

int main() {
    freopen("out.txt", "w", stdout);
    // for (double eeps = 3; eeps <= 11; eeps += 0.1) {
    for(int M = 50; M != 201; ++M){
    eps = pow(10, -5);
    int now = 0, pre = 1;
    for (int i = 1; i <= M; i++)
        for (int j = 1; j <= M; j++)
        if (i == 1)
            a[now][i][j] = 1;
        else if (j == 1 || j == M || i == M)
            a[now][i][j] = 0;
        else
            a[now][i][j] = 0.5;
    int I = 0, flag = 0;
    for (; !flag;) {
        flag = 1, now ^= 1, pre ^= 1;
        for (int i = 1; i <= M; i++)
        for (int j = 1; j <= M; j++)
            if (i == 1)
            a[now][i][j] = a[pre][i][j];
            else if (j == 1 || j == M || i == M)
            a[now][i][j] = a[pre][i][j];
            else
            a[now][i][j] = a[pre][i - 1][j] + a[pre][i + 1][j] +
                           a[pre][i][j - 1] + a[pre][i][j + 1],
            a[now][i][j] /= 4;
        for (int i = 1; i <= M; i++)
        for (int j = 1; j <= M; j++)
```

```
                if (fabs(a[now][i][j] - a[pre][i][j]) / a[now][i][j]
                    > eps) {
                flag = 0;
                break;
                }
            I++;
        }
        printf("%d %d\n", M, I);
        }
        return 0;
}
```