Using the Relaxation Method to solve Poisson's Equation

Nicole Nikas

October 16, 2015

Abstract

In this paper I solve Poisson's equation using a combination of algorithms. I will use the Relaxation Method, the Jacobi Iteration, and the Gauss-Seidel adaptation to the Jacobi Iteration. I used these methods to develop a numerical solution to the electric potential at any given point within a two dimensional boundary of N lattice points.

1 Poisson's Equation in Electrostatics

Poisson's Equation for electrostatics is derived using Gauss Law. We start with the divergence of the Electric Field. This is equal to the charge density over the permittivity. The Electric Field is the equal to the negative divergence of the electric potential. Therefore we end up with the divergence of the negative gradient of the electric potential is shown bellow using the bellow equation.

$$\nabla \cdot (-\nabla \phi) = -\rho/\epsilon \tag{1}$$

This can be shown using the Laplace operator as

$$\nabla^2 = -\rho/\epsilon \tag{2}$$

Poisson's equation is an elliptic partial differential equation of the second form. Therefore the electric potential in two dimensions, can be written as:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\phi(x,y) = -\rho(x,y)/\epsilon \tag{3}$$

2 The Relaxation Method

The relaxation method is an iterative methods used for solving systems of equations. This means they generate a sequence of numbers that update the approximate solutions for a specific problem. It's called the relaxation method because that is exactly what it does, it smooths out the fine-scale factors and generalizes a solution.

3 The Relaxation Method and Laplace's Equation

I will explain the usefulness of the Relaxation Method with Laplace's Equation. We first need to declare ρ the charge density. We do this by setting the value of the potential at the surface, which is a first-type boundary condition.

This is where the relaxed formula for Poisson's equation is derived from, the averaging of the four exterior points is the potential at that point. It is given by the notations that $\phi - i, j$ at any given point where i and j are the x and y positions. We use h as a spacing constant. It is equal to $1/(1+n)$. The formula is:

$$\rho_{i,j} = \left(\frac{h^2}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}\right) \tag{4}$$

which can also be written as:

$$V_{i,j} = \frac{1}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4h^2\rho_{i,j}) \tag{5}$$

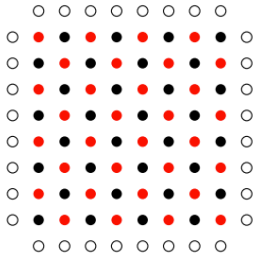

Figure 1: This gives us a two dimensional plane with set boundaries with N^2 dots, in this case N is 9

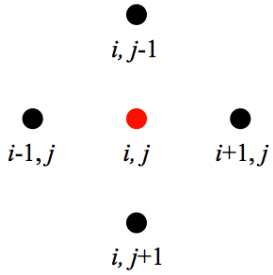

Figure 2: The values of the red dots are averaged by the 4 values of their surrounding black dots shown in the above figure

To optimize our solution we introduce the over-relaxation parameter that diverges the fastest when it is set to:

$$\omega = \frac{2}{1 + \frac{\pi}{N}} \tag{6}$$

so equation 4 now becomes:

$$\phi_{i,j} = \Big(1 - \omega)\phi_{i,j} \frac{\omega}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4h^2\rho_{i,j}) \tag{7}$$

4 The Jacobi and Gauss-Seidel Extension

For a given charge density, we can find the electric potential at any point by determine the charge from the 4 surrounding points. We can then use the Jacobi Iteration to help compute the final relaxed formula. This helps solve the equation because we are solving for the right and left hand side of the equation simultaneously. We do this by starting with a guess for the initial value of the electric potential. We plug this in as the electric potential new value which is going to be on the right hand side. We then set this to the same equation as a above, except now we are using old and new values of the electric potential to help determine the next electric potential. It is shown as follows(ignoring the over relaxation constant for now):

$$\phi_{i,j}^{n+1} = \phi_{i,j} \frac{1}{4}(\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 4h^2\rho_{i,j}) \tag{8}$$

This process is repeated until it converges to the correct solution. We can then update the Jacobi iteration with the Gauss-Seidel iteration which converges a little faster. It does this by observing the pattern that the notes are always visited in sequence. It also uses the values that are recently updated as soon as their available. The Gauss-Seidel iteration looks like this:

$$\phi_{i,j}^{n+1} = \phi_{i,j} \frac{1}{4}(\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n+1} - 4h^2\rho - i, j) \tag{9}$$

We then add the over-relaxation constant back, and we have the Successive Over-relaxation iteration(SOR).

$$\phi_{i,j}^{n+1} = \phi_{i,j} \frac{\omega}{4}(\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n+1} - 4h^2\rho - i, j) \tag{10}$$

This is where the computational part of this problem comes into play, as we can graph the value of the electric potential between the boundaries(where $\phi = 0$) of the x and y plane for updated values of i and j.

5 An Example

I will give an example to illustrate the calculation. To simplify things we will exclude ρ, and instead set a second type of boundary condition by setting the electric potential $\phi_{i,j}$ to certain values for when i and j are 0 or before they reach the maximum number of lattice points $N - 1$. We will also exclude ht spacing constant for simple computation. Our final equation excluding rho is:

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} + \frac{\omega}{4}(\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n+1} - 4\phi_{i,j}^{n}) \tag{11}$$

The rest is just updating the old values of $\phi(i, j)$ by the old and new surrounding points and averaging them. We compute this code using the C++ code in the next section. The code runs until an acceptable error is met. We calculate this by determining a value of the maximum error which is

$$ErrMax = \frac{\phi_{i,j}^{n} + 1 - \phi_{i,j}^{n}}{\phi_{ref}} \tag{12}$$

Where ϕ_{ref} is a given reference value. Iterations of the SOR stop once this error has met the tolerance level, for an example of 10^{-5}

If we wanted to simplify this equation one step further we would exclude omega and the first and last factor so that it is simplifying averaging the four points and updating that value to the new point. This equation looks like this:

$$\phi_{i,j}^{n+1} = \frac{1}{4}(\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n+1}) \tag{13}$$

Both equations come to the same conclusion, however the first takes less steps to compute it. For example if we are computing between boundaries of 1, 0, 0, and 0. The error is the same, $1 * 10^{-}6$, however the first equation ran it in 187 steps as apposed to the second equation that ran the same error in 3,660 steps. Therefore the first equation is more efficient of an update.

6 Examples of Graphs

All of the bellow graphs are using 100 lattice points.

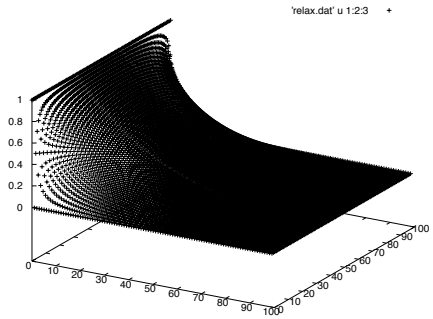

Figure 3: This graph uses boundaries of 1, 0, 0, 0

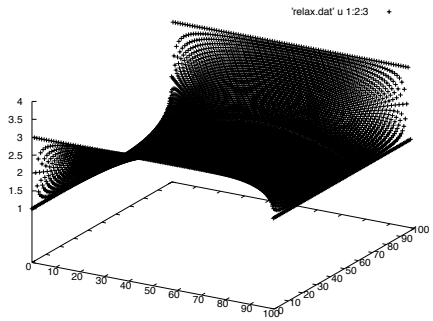

Figure 4: This graph uses boundaries of 1, 2, 3, 4

7 Conclusion

Finally, we see that the solution to the partial differential equation presented by Poisson's equation can be solved using a combination of methods to develop a numerical solution. You can do this by combining the Relaxation Method, the Jacobi iteration and Gauss-Seidel iteration. We were then able to determine a formula for the value of the electric potential,ϕ, at a given point based on the given boundary conditions. Next step in the computation was developing a C++ program that could run over a certain number of iterations, with a maximum error less than the tolerance allowed. This gives us a final solution for the Poisson's equation that is the most approximately accurate.

8 Sources

E. M. Purcell, *Electricity and magnetism*, McGraw-Hill, (1985).

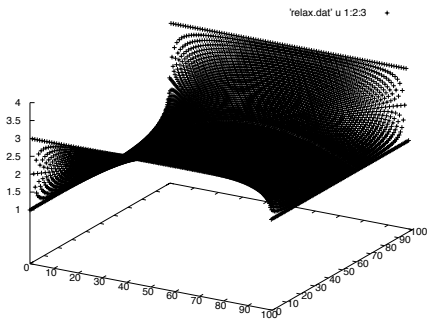

Figure 5: This graph uses boundaries of 50, 0, 50, 0

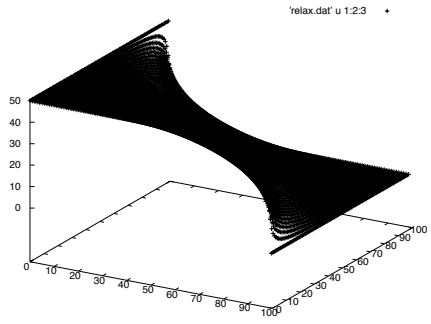

Figure 6: This graph uses boundaries of 50, 0, 50, 0

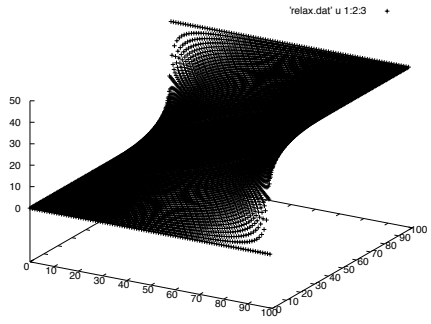

Figure 7: This graph uses boundaries of 0, 50, 0, 50

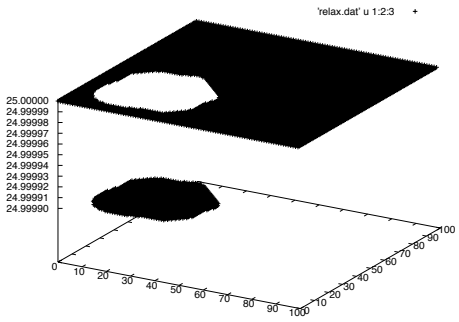

Figure 8: This graph uses boundaries of 25, 25, 25, 25

C. Aubin, *Lecture Notes from Computational Physics*,(2015).
M. Greenberg, *Advanced Engineering Mathematics*, 2nd Edition, Pearson (1998).
Richard. J. Gonzales,*Lecture Notes from Computational Physics*,(2008).