

# Computing Electrostatic Potentials on a Grid Using Successive Over-Relaxation (SOR)

Ross L. Spencer

Brigham Young University

Most of the electrostatic problems that come up in the real world are too hard to solve with formulas, or even by Fourier series. For such problems the most widely used method is to use a spatial grid. This can be done in 1, 2, or 3 dimensions, but the technique will be illustrated here in 2 dimensions.

## The Basic Algorithm

Suppose that you have an electrostatics problem in infinitely-long geometry so that one of the dimensions (say the  $z$  dimension) is irrelevant. Then the equation that must be solved is Poisson's equation in the two dimensions  $x$  and  $y$ :

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{\rho}{\epsilon_0} \quad , \quad (1)$$

subject to boundary conditions on  $V$ . To solve this problem on a grid we first choose a rectangular computation region bounded in  $x$  by  $x_{min}$  and  $x_{max}$  and in  $y$  by  $y_{min}$  and  $y_{max}$ . We then subdivide the  $x$  interval into  $N_x$  subintervals and the  $y$  interval into  $N_y$  subintervals to form a grid in the  $xy$  plane. The  $x$ -subintervals have length  $\Delta x = (x_{max} - x_{min})/N_x$  and the  $y$ -subintervals have length  $\Delta y = (y_{max} - y_{min})/N_y$ . Because the ends of the intervals are included the grid is of size  $(N_x + 1) \times (N_y + 1)$  and the  $(x_i, y_j)$  position of the gridpoint labeled by  $(i, j)$  is given by

$$x_i = x_{min} + (i - 1)\Delta x \quad ; \quad y_j = y_{min} + (j - 1)\Delta y \quad . \quad (2)$$

We now use this grid to solve Poisson's equation by writing down an approximation to it at each grid point. The approximation is obtained by using the centered-difference approximation to the second derivative:

$$\frac{\partial^2 V}{\partial x^2} \Big|_{(i,j)} \approx \frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{\Delta x^2} \quad . \quad (3)$$

where  $V_{i,j} = V(x_i, y_j)$ . This approximation gives the following grid version of Poisson's equation

$$\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{\Delta x^2} + \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{\Delta y^2} = -\frac{\rho_{i,j}}{\epsilon_0} \quad . \quad (4)$$

It is by no means clear that this approximation is useful just by looking at it, but a large number of clever people have discovered over the years that it is useful to rewrite this equation by solving for  $V_{i,j}$ :

$$V_{i,j} = \left( \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} \right)^{-1} \left[ \frac{V_{i+1,j} + V_{i-1,j}}{\Delta x^2} + \frac{V_{i,j+1} + V_{i,j-1}}{\Delta y^2} + \frac{\rho_{i,j}}{\epsilon_0} \right] \quad . \quad (5)$$

The reason that this is a good idea is that it turns out that in this form the equation can be solved by iteration, just as we can solve the equation  $x = \cos x$  by iteration:

$$x_{n+1} = \cos x_n \quad , \quad (6)$$

where  $n$  counts successive iterations. If you haven't seen this trick before, try it. Start with 1, then calculate  $\cos(1)$ , then  $\cos(\cos(1))$ , etc.. After doing this a bunch of times you will get the number 0.7391, which solves the equation  $x = \cos x$ . The same trick works for Eq. (5): make a guess for  $V_{i,j}$  across the whole grid, compute the right hand side of Eq. (5) from this guess and use the equation to get a new set of values for  $V_{i,j}$ . Rewriting Eq. (5) in the symbolic form

$$V = Rhs(V) \quad , \quad (7)$$

this iteration scheme can be written as

$$V_{n+1} = Rhs(V_n) \quad . \quad (8)$$

Amazingly, this actually works and is called Successive Relaxation, as discussed on page 114 of Griffiths. It is, however, painfully slow. But the people who discovered this trick also discovered how to improve it in various ways and have, in fact, developed an entire field of study based on this problem. We will just discuss here the simplest improvement that is useful, but if you would like to see why this really works and be introduced to even better methods, take our new Computational Physics course Physics 512, taught every fall.

The simple, but effective, improvement we will use is called Successive Over-Relaxation, or SOR for short, and it simply modifies the iteration scheme given above in the following way:

$$V_{n+1} = \omega Rhs(V_n) + (1 - \omega)V_n \quad , \quad (9)$$

where  $\omega$  is a number between 1 and 2. Using  $\omega = 1$  is just simple relaxation and using  $\omega = 2$  makes the iteration unstable so that  $V$  goes to infinity. But between these two extremes there is an optimum value of  $\omega$  that makes the iteration converge much better than simple relaxation. You can get close to the best value of  $\omega$  by experimenting, but here is some rough guidance. For a  $20 \times 20$  grid use  $\omega = 1.7$ ; for a  $30 \times 30$  grid use  $\omega = 1.8$ ; for a  $40 \times 40$  grid use  $\omega = 1.85$ ; and for a  $50 \times 50$  grid use  $\omega = 1.9$ .

## The Boundary Conditions

Oh, we forgot about the boundary conditions. This iteration scheme works great at points away from metal conductors, but what are we supposed to do at points where the potential is specified? The answer is easy: set  $V$  at these points to what it is supposed to be and then don't ever change them. And that's all there is to it. You will have some homework problems to do using this technique, but to make it easier you will be given working code which you can modify to solve the assigned problems.

## Finding $V$ and $\mathbf{E}$ at Arbitrary Points $(x, y)$

This works fine for finding  $V(x, y)$  at the defined grid points, but what if you want  $V(x, y)$  at some arbitrary point? Well, then you have to interpolate in two dimensions. There are lots of ways to do this, but the simplest (and usually adequate, but least accurate) way is bilinear interpolation. In this scheme you begin by finding the four grid points that surround the point  $(x, y)$  at which you want to find the potential  $V(x, y)$ . In what follows these points will be designated by the symbols 00, 10, 01, and 11 denoting, respectively, the lower left point, the lower right point, the upper right point, and the upper left point. The bilinear interpolation formula is

$$\begin{aligned}
V(x, y) \approx & [1 - (x - x_{00})/\Delta x][1 - (y - y_{00})/\Delta y]V_{00} + [(x - x_{00})/\Delta x][1 - (y - y_{00})/\Delta y]V_{10} \\
& + [(x - x_{00})/\Delta x][(y - y_{00})/\Delta y]V_{11} + [1 - (x - x_{00})/\Delta x][(y - y_{00})/\Delta y]V_{01} \quad (10)
\end{aligned}$$

This formula takes a weighted average of the neighboring points to estimate  $V(x, y)$  inside the region bounded by these grid points.

Now suppose you want not just  $V(x, y)$ , but also  $\mathbf{E}(x, y)$ ; what do you do then? First you have to find  $\mathbf{E}$  at all the grid points, and then you can use bilinear interpolation on the  $x$  and  $y$ -components of  $\mathbf{E}$  just as it was described for  $V$ . To find the components of  $\mathbf{E}$  recall that

$$\mathbf{E} = -\nabla V = \mathbf{i} \frac{\partial V}{\partial x} - \mathbf{j} \frac{\partial V}{\partial y} \quad (11)$$

So we just need to know how to numerically compute these partial derivatives using values of  $V$  on the grid. This is most easily accomplished by using a simple centered difference approximation:

$$\left. \frac{\partial V}{\partial x} \right|_{i,j} \approx \frac{V_{i+1,j} - V_{i-1,j}}{2\Delta x} \quad ; \quad \left. \frac{\partial V}{\partial y} \right|_{i,j} \approx \frac{V_{i,j+1} - V_{i,j-1}}{2\Delta y} \quad (12)$$