Solving Laplace's Equation

Written by Dave Pawlowski

Introduction

Unlike ordinary differential equations, which can be solved using general techniques such as Euler's method or a Runge-Kutta schemes, general numerical solutions to partial differential equations (PDEs) do not exist. Instead, there are a wide range of numerical techniques that can be applied to different classes of PDEs. The goal here is to numerically solve one such class referred to as elliptic PDEs, one example of such an equation being Laplace's equation, which describes the electric potential in space in absence of any electric charges:

$$\nabla^2 V = 0 \tag{1}$$

or

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

To solve this equation, we will focus on the use of numerical schemes called relaxation methods.

To start, it is necessary to discretize the independent variables, x, y, and z. In other words, we create a grid for which we will determine the potential $V(i,j,k)$, where i,j,k refer to the indices of a particular grid point. Then, we need to rewrite the derivatives in Eq. 2 in terms of a finite difference. We begin with

$$\frac{\partial V}{\partial x} \approx \frac{V(i+1,j,k) - V(i,j,k)}{\Delta x} \tag{3}$$

where we have used forward differencing. We could have just have easily used the backward or central difference. Note that Eq. 3 is effectively centered at the point $i + \frac{1}{2}$. We can then approximate the second derivative of the function as

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{1}{\Delta x}\left[\frac{\partial V}{\partial x}(i+\frac{1}{2}) - \frac{\partial V}{\partial x}(i-\frac{1}{2})\right], \tag{4}$$

where it is explicit that the 1st derivatives are really evaluated at $i \pm \frac{1}{2}$. Putting it all together, we have

$$\frac{\partial^2 V}{\partial x^2} \approx \left[\frac{V(i+1,j,k) - V(i,j,k)}{\Delta x} - \frac{V(i,j,k) - V(i-1,j,k)}{\Delta x}\right] \tag{5}$$

$$\approx \frac{V(i+1,j,k) + V(i-1,j,k) - 2V(i,j,k)}{(\Delta x)^2}. \tag{6}$$

This equation says that in order to determine the function at the grid point that we are interested in, $V(i,j,k)$, we must know the function at the surrounding grid points $i+1, i-1$, etc. In other

words, the value of V at a particular grid point is the average of V at all the neighboring points. The solution, $V(x,y,z)$, then is a function that satisfies that condition at all points simultaneously.

Generalizing this problem to 3-dimensions follows the same steps as above and results in

$$V(i,j,k) = \frac{1}{6}[V(i+1,j,k) + V(i-1,j,k) + V(i,j+1,k) + V(i,j-1,k) + V(i,j,k+1) + V(i,j,k-1)]], \tag{7}$$

where we have rearranged and solved for the point of interest. Note that the factor of $\frac{1}{6}$ comes from the 3-D geometry. If this were a 2-D problem, the factor would be $\frac{1}{4}$. There is an issue here though. In order to solve this problem, we need to have prior information about the function at all grid points, not just the boundary. The solution to this is to start with a guess of the function. In many cases, it is sufficient to guess $V = 0$ everywhere (except on the boundary). To improve our guess, we evaluate Eq 7 at each point on our grid using our initial guess for the function in all terms on the RHS of the equation. Once we have solved for $V(i,j,k)$ at all points on our grid, we can repeat the procedure using our updated guess to fill in the RHS. In that way, we *relax* our initial guess into the solution (to some confidence level) by iterating over the grid, updating the RHS of the equation with our latest and greatest guess at the function. This processes is referred to as the **Jacobi method** or **Jacobi Relaxation**.

This method works due to the uniqueness theorem of Laplace's equation, which basically states that if you find a solution that satisfies the equation, then it is the one and only solution. That means that given information about the boundary conditions, we are free to simply guess the solution to Laplace's equation within our grid. If we guess a solution that works, then we are done. Jacobi relaxation simply amounts to 'strategically guessing'.

Implementation of Jacobi Relaxation is relatively straight forward. Setup a grid and start with an initial guess for the value of $V(i,j,k)$ on the grid while also including the boundary conditions. Update $V(i,j,k)$ repeatedly for each grid point using Eq. 7 while simultaneously checking for convergence. Convergence is achieved when the change to $V(i,j,k)$ is sufficiently small (based on the problem).

A note about relaxation techniques

The use of relaxation in this way is in many ways similar to taking a time-independent problem and turning it to a time dependent one. Imagine a toy problem where we know the potential of some 2D region of space at the boundary: $V(0,y) = 1$ and $V(1,y) = -1$. We wish to find the potential everywhere inside the domain $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$. Jacobi relaxation essentially solves the time dependent problem where at $t < 0$, $V = 0$ everywhere, then at $t = 0$ the boundary conditions are suddenly applied (i.e. two conducting walls are held at +/- 1V.). At exactly $t = 0$, the potential is zero everywhere except at the walls. As time evolves, the potential between the walls change to accommodate the boundary conditions, and after some finite amount of time (a very, very small finite amount of time since the electric field is propagating at the speed of light), the potential inside the domain is finite everywhere and reaches a steady-state. Each iteration of the Jacobi method gives a snapshot of the potential before that 'steady-state' value is reached, but given enough time, the solution relaxes to a steady, unchanging, value.

Higher-order methods

Generally, Jacobi Relaxation is a very conservative, and thus inefficient, algorithm. In fact, if a toy 2D problem has L points on each side, then the number of iterations required for a given level of convergence goes as L^2. In other words, if we increase the number of grid points by a factor of 2, then the computational effort is increased by a factor of 4. We can get a small amount of improvement by using new information about the function as it becomes available. For example, Jacobi Relaxation for a 2-D problem can be expressed in slightly more detail as:

$$V_{new}(i,j) = \frac{1}{4}[V_{old}(i+1,j) + V_{old}(i-1,j) + V_{old}(i,j+1) + V_{old}(i,j-1)]. \tag{8}$$

Here V_{old} is the potential from the previous guess. A different method, the Gauss-Seidel method uses the new values of V as they become available, depending on how you loop through the domain. For example, if you from small i, j to large i, j, then

$$V_{new}(i,j) = \frac{1}{4}[V_{old}(i+1,j) + V_{new}(i-1,j) + V_{old}(i,j+1) + V_{new}(i,j-1)]. \tag{9}$$

Employing the Gauss-Siedel method improves the speed of convergence by a factor of 2, which is nice, but not great. A more widely used method that can improve convergence speed by an order of L is the method of *simultaneous over-relaxation* (SOR), which uses a problem-specific over-relaxation factor that maximizes the efficiency of the scheme. There is plenty of discussion on how to implement this method in the literature so it will not be discussed here. However, the basic premise is much the same as that of Jacobi Relaxation and the Gauss-Siedel method.