

Topic 4: Multigrid Methods: Poisson's Equation

A *multigrid method* for solving elliptic partial differential equations was introduced by A. Brandt, *Mathematics of Computation* **31**, 333 (1977). He showed how an elliptic PDE discretized on N grid points could be solved in time of $\mathcal{O}(N)$, which is a huge improvement on the $\mathcal{O}(N^2)$ time required by the classical Jacobi method.

Poisson's Equation

Poisson's equation in electrostatics

$$\nabla^2 \Phi(\vec{r}) = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} = -\frac{\rho(\vec{r})}{\epsilon_0}.$$

determines the electrostatic potential $\Phi(\vec{r})$ due to a given charge distribution $\rho(\vec{r})$.

Poisson's equation is an example of an *elliptic partial differential equation*. A differential operator of the type $a^2 \partial^2 / \partial x^2 + b^2 \partial^2 / \partial y^2 + c^2 \partial^2 / \partial z^2$ is said to be elliptic because its action on the periodic function $\exp(i\vec{k} \cdot \vec{r})$ yields the function $a^2 k_x^2 + b^2 k_y^2 + c^2 k_z^2$, and the surfaces in \vec{k} -space on which this function has constant values are the surfaces of ellipsoids.

It can be shown that the solution of an elliptic equation is uniquely determined throughout the interior of some *closed* region \mathcal{R} , which can be multiply connected (i.e., can have holes in it) if one of the following types of boundary conditions are specified on the surface \mathcal{S} of the region \mathcal{R} :

1. Dirichlet boundary conditions, in which the value of the function Φ is specified on \mathcal{S} , or
2. Neumann boundary conditions, in which the *normal derivative* $\hat{n} \cdot \vec{\nabla} \Phi$, \hat{n} being the outward normal to the surface, is specified on \mathcal{S} , or
3. periodic boundary conditions.

Dirichlet boundary conditions are appropriate when the potential (voltage) is fixed on the bounding surface of the region, and Neumann boundary conditions are appropriate when the normal derivative of the potential, i.e., the electric field component normal to the bounding surface is specified. Poisson's equation also determines the Newtonian gravitational potential due to a distribution of masses.

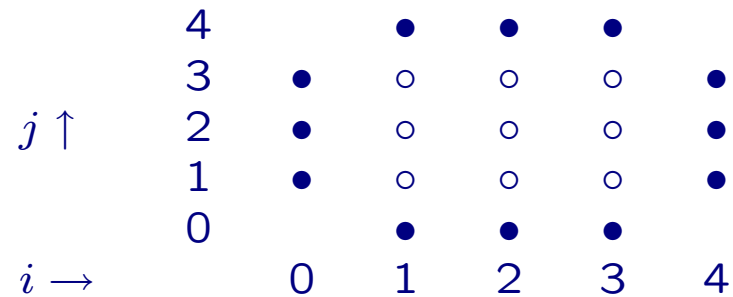
Discretizing Poisson's equation in 2-D

Let's consider Poisson's equation in two dimensions

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = -\frac{\rho(x, y)}{\epsilon_0}.$$

Suppose, for simplicity, that we are required to solve this equation in the interior of a square in x - y space. We introduce a discrete rectangular grid of points which form a spatial lattice. Let us also assume that the lattice spacings in the x and y directions

are equal: $\delta x = \delta y = h$. The lattice sites can be labeled by a pair of integers (i, j) as in the following diagram



in which the solid circles represent boundary points on the closed boundary, and the open circles represent interior points. If (x_0, y_0) is the lower left-hand corner of the square, then the values of Φ at the lattice sites can be represented by $\phi_{ij} = \Phi(x_0 + ih, y_0 + jh)$. Since Poisson's equation is an elliptic equation, we shall obtain a unique solution at the interior points if the values of Φ at the boundary points are specified. In the example shown in the figure, there are 12 boundary points at which the values of ϕ_{ij} are assumed specified, and 9 interior points at which the values of ϕ_{ij} are to be found by solving Poisson's equation. A simple discretization is

$$\frac{\phi_{i+1,j} + \phi_{i-1,j} - 2\phi_{ij}}{h^2} + \frac{\phi_{i,j+1} + \phi_{i,j-1} - 2\phi_{ij}}{h^2} = -\frac{\rho_{ij}}{\epsilon_0},$$

where $\rho_{ij} = \rho(x_0 + ih, y_0 + jh)$. We can write down 9 equations of this form in which i and j independently take values $1 \dots 3$, i.e., there is one equation for each of the interior points. These 9 equations must then be solved for the 9 unknown ϕ_{ij} values at the interior points.

Matrix nature of Poisson's equation.

It is easy to see that linear partial differential equations can be represented using matrices when they are discretized. Thus in principle one can use the methods and algorithms of linear algebra to solve such equations. In practice, the number of equations to be solved simultaneously can be very large!

Let's consider the 9 equations in the 2-D example discussed above. The equation at the site $i = 1, j = 1$, for example, is

$$\frac{\phi_{21} + \phi_{01} - 2\phi_{11}}{h^2} + \frac{\phi_{12} + \phi_{10} - 2\phi_{11}}{h^2} = -\frac{\rho_{11}}{\epsilon_0} .$$

Since ϕ_{01} and ϕ_{10} are fixed boundary values, this equation can be rewritten in the form

$$-4\phi_{11} + \phi_{21} + \phi_{12} = -\frac{h^2\rho_{11}}{\epsilon_0} - \phi_{10} - \phi_{01} \equiv \sigma_{11} ,$$

where σ_{11} is a known quantity. As a second example, the equation at site (2, 2) is

$$\phi_{21} + \phi_{12} - 4\phi_{22} + \phi_{32} + \phi_{23} = -\frac{h^2\rho_{22}}{\epsilon_0} \equiv \sigma_{22} .$$

The other 7 equations can be rewritten in a similar way. The set of 9 equations, and their solution, can then be expressed in matrix form:

$$M\phi = \sigma \quad \Rightarrow \quad \phi = M^{-1}\sigma ,$$

with

$$\phi = \begin{pmatrix} \phi_{11} \\ \phi_{21} \\ \phi_{31} \\ \phi_{12} \\ \phi_{22} \\ \phi_{32} \\ \phi_{13} \\ \phi_{23} \\ \phi_{33} \end{pmatrix}, \quad M = \begin{pmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{pmatrix}, \quad \sigma = \begin{pmatrix} \sigma_{11} \\ \sigma_{21} \\ \sigma_{31} \\ \sigma_{12} \\ \sigma_{22} \\ \sigma_{32} \\ \sigma_{13} \\ \sigma_{23} \\ \sigma_{33} \end{pmatrix}.$$

The matrix M in this equation is said to be *tridiagonal with fringes*. It is also an example of a *sparse matrix*, i.e., one in which most of the matrix elements are zero.

Efficient algorithms for inverting such matrices are required, especially in three or more spatial dimensions: for example, a lattice with 100 points in each of the x , y and z directions has 10^6 points, and one is faced with the problem of finding the inverse of a $10^6 \times 10^6$ matrix! Algorithms for inverting matrices are discussed for example in Chapter 2 Solution of Linear Algebraic Equations of *Numerical Recipes*. The most efficient algorithms take time of $\mathcal{O}(N^{3/2})$: if the number of equations is $\sim 10^6$, the run time will be $\sim 10^9$.

The matrix inverse method for solving a discretized PDE is an example of a *direct method*, i.e., one in which the solution is obtained, with an accuracy limited only by the discretization approximation and roundoff errors, in a single step.

Iterative methods: Jacobi, Gauss-Seidel, SOR

In contrast to direct matrix methods, which require sophisticated linear algebra algorithms, there are iterative methods for solving elliptic PDE's that are very simple to understand and implement.

The classical iterative methods were introduced by the famous mathematicians Jacobi and Gauss. Let's discuss their methods for the 2-D Poisson equation.

The exact solution to the discretized problem obeys the equation

$$\phi_{ij} = \frac{1}{4} \left[\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} + h^2 \frac{\rho_{ij}}{\epsilon_0} \right].$$

This equation cannot be solved explicitly for fixed i, j because there are 5 unknowns involved. However, suppose we were to guess approximate values for the 4 neighboring ϕ 's on the right hand side, then if this equation were to give us an improved estimate of ϕ_{ij} , the process can be iterated until it converges to the exact solution. Thus if the n -th iterate is denoted ϕ_{ij}^n , Jacobi's algorithm is

$$\text{Jacobi: } \phi_{ij}^{n+1} = \frac{1}{4} \left[\phi_{i+1,j}^n + \phi_{i-1,j}^n + \phi_{i,j+1}^n + \phi_{i,j-1}^n + h^2 \frac{\rho_{ij}}{\epsilon_0} \right].$$

The algorithm is primed with an initial guess ϕ_{ij}^0 , and the solution is iterated until some convergence criterion is satisfied. It can be shown that the Jacobi method is stable and converges: the number of iterations to reduce the error in the solution by a factor of 10^{-p} is $\sim \frac{1}{2}pN$, where N is the number of lattice points. Since each iteration

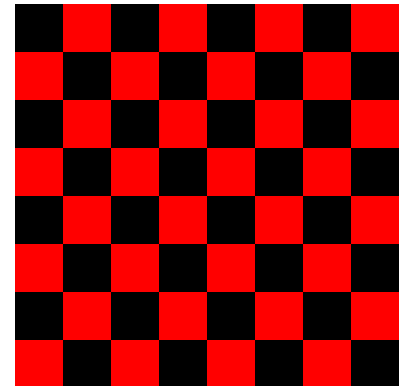
involves updating N value of ϕ , the algorithm scales like $\mathcal{O}(N^2)$. This is very slow for large N .

The Jacobi algorithm can be speeded up by a factor of 2 using

$$\text{Gauss-Seidel: } \phi_{ij}^{n+1} = \frac{1}{4} \left[\phi_{i+1,j}^n + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^n + \phi_{i,j-1}^{n+1} + h^2 \frac{\rho_{ij}}{\epsilon_0} \right].$$

Here we assume that the lattice is swept with i and j increasing so that we can use the most recently updated values of ϕ in the iteration. It can be shown that the number of iterations to reduce the error in the solution by a factor of 10^{-p} is $\sim \frac{1}{4}pN$, which is an improvement over Jacobi, but does not improve the overall $\mathcal{O}(N^2)$ time requirement. There are variations on the Gauss-Seidel formula depending on the order in which ϕ_{ij} values are updated.

Numerical Recipes recommends using *red-black* ordering in making a Gauss-Seidel pass through the lattice: make one pass updating the “even” points (like the red squares on a checkerboard), and then a second pass updating the “odd” points (like the black squares), where “odd” and “even” refer to the sum $i+j$ of lattice indices. Note that the four nearest neighbors of a red square are black, and vice versa. Checkerboard updating avoids the directional asymmetry caused by simple Gauss-Seidel updating.



The *Successive Overrelaxation* (SOR) algorithm uses a linear combination, determined by an *overrelaxation parameter* ω , of the old and improved iterates

$$\text{SOR: } \phi_{ij}^{n+1} = (1 - \omega)\phi_{ij} + \frac{\omega}{4} \left[\phi_{i+1,j}^n + \phi_{i-1,j}^n + \phi_{i,j+1}^n + \phi_{i,j-1}^n + h^2 \frac{\rho_{ij}}{\epsilon_0} \right].$$

This formula has been written with the Jacobi iterate on the right hand side: however, the algorithm can also be used with Gauss-Seidel updating (either the asymmetric or checkerboard versions). The following results can be proved:

- The iteration converges only if $0 < \omega < 2$.
- If $1 < \omega < 2$ the iteration converges faster than the $\omega = 1$ version, i.e., straight Jacobi or Gauss-Seidel.
- There is an optimum value of ω for which the convergence is fastest. For a square lattice in 2-D

$$\omega_{\text{opt}} \simeq \frac{2}{1 + \frac{\pi}{\sqrt{N}}},$$

where \sqrt{N} is the number of lattice points in each direction.

The remarkable fact about the SOR algorithm is that the number of iterations to reduce the error in the solution by a factor of 10^{-p} can be shown to be $\sim \frac{1}{3}p\sqrt{N}$ if $\omega = \omega_{\text{opt}}$. This implies that SOR has run time of $\mathcal{O}(N^{3/2})$, which is comparable with the most efficient direct matrix inversion algorithms!

Thus successive overrelaxation was the method of choice for solving elliptic PDE's until Brandt discovered the multigrid algorithm, which can solve PDE's in $\mathcal{O}(N)$ running time, about 25 years ago.