Topic 4: Multigrid Methods: Poisson's Equation

The multigrid method provides algorithms which can be used to accelerate the rate of convergence of iterative methods, such as Jacobi or Gauss-Seidel, for solving elliptic partial differential equations.

Iterative methods start with an approximate guess for the solution to the differential equation. In each iteration, the difference between the approximate solution and the exact solution is made smaller.

One can analyze this difference or *error* into components of different wavelengths, for example by using Fourier analysis. In general the error will have components of many different wavelengths: there will be short wavelength error components and long wavelength error components.

Algorithms like Jacobi or Gauss-Seidel are *local* because the new value for the solution at any lattice site depends only on the value of the previous iterate at neighboring points. Such local algorithms are generally more efficient in reducing short wavelength error components.

The basic idea behind multigrid methods is to reduce long wavelength error components by updating blocks of grid points. This strategy is similar to that employed by cluster algorithms in Monte Carlo simulations of the Ising model close to the phase transtion temperature where long range correlations are important. In fact, multigrid algorithms can also be used in Monte Carlo simulations: we will study an application to the Monte Carlo simulation of the 2-D $X-Y$ model later, but for now let's consider the 2-D Poisson equation of electrostatics, which is a typical elliptic PDE.

Multigrid solution of Poisson's equation in 2-D

With a small change in notation, Poisson's equation in 2-D can be written:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f(x,y) \ ,$$

where the unknown solution $u(x,y)$ is determined by the given *source term* $f(x,y)$ in a closed region. Let's consider a square domain $0 \leq x, y \leq L$ with homogeneous Diriclet boundary conditions $u = 0$ on the perimeter of the square. The equation is discretized on a grid with $N+2$ lattice points, i.e., N interior points and 2 boundary points, in the x and y directions. At any interior point, the exact solution obeys

$$u_{i,j} = \frac{1}{4}\left[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} + h^2 f_{i,j}\right] \ .$$

The algorithm uses a succession of lattices or grids. The number of different grids is called the number of multigrid levels ℓ. The number of interior lattice points in the x and y directions is then taken to be 2^ℓ, so that $N = 2^\ell + 2$, and the lattice spacing $h = 1/(N-1)$. N is chosen in this manner so that the downward multigrid iteration can construct a sequence of coarser lattices with

$$2^{\ell-1} \rightarrow 2^{\ell-2} \rightarrow \ldots \rightarrow 2^0 = 1$$

interior points in the x and y directions.

Suppose that $u(x,y)$ is the approximate solution at any stage in the calculation, and $u_{\text{exact}}(x,y)$ is the exact solution which we are trying to find. The multigrid algorithm uses the following definitions:

- The *correction*

$$v = u_{\text{exact}} - u$$

  is the function which must be added to the approximate solution to give the exact solution.

- The *residual* or *defect* is defined as

$$r = \nabla^2 u + f \ .$$

Notice that the correction and the residual are related by the equation

$$\nabla^2 v = \left[\nabla^2 u_{\text{exact}} + f\right] - \left[\nabla^2 u + f\right] = -r \ .$$

This equation has exactly the same form as Poisson's equation with v playing the role of unknown function and r playing the role of known source function!

Simple $V-$cycle algorithm

The simplest multigrid algorithm is based on a *two-grid* improvement scheme. Consider two grids:

- a *fine grid* with $N = 2^{\ell} + 2$ points in each direction, and

- a *coarse grid* with $N = 2^{\ell-1} + 2$ points.

We need to be able to move from one grid to another, i.e., given any function on the lattice, we need to able to

- *restrict* the function from fine $\rightarrow$ coarse, and
- *prolongate* or *interpolate* the function from coarse $\rightarrow$ fine.

Given these definitions, the multigrid $V-$cycle can be defined recursively as follows:

- If $\ell = 0$ there is only one interior point, so solve exactly for
$$u_{1,1} = (u_{0,1} + u_{2,1} + u_{1,0} + u_{1,2} + h^2 f_{1,1})/4 \; .$$
- Otherwise, calculate the current $N = 2^\ell + 2$.
- Perform a few pre-smoothing iterations using a local algorithm such as Gauss-Seidel. The idea is to damp or reduce the short wavelength errors in the solution.
- Estimate the correction $v = u_{\text{exact}} - u$ as follows:
  - Compute the residual
$$r_{i,j} = \frac{1}{h^2}\left[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\right] + f_{i,j} \; .$$
  - Restrict the residual $r \rightarrow R$ to the coarser grid.

  - Set the coarser grid correction $V = 0$ and improve it recursively.
  - Prolongate the correction $V \to v$ onto the finer grid.

- Correct $u \to u + v$.

- Perform a few post-smoothing Gauss-Seidel interations and return this improved u.

How does this recursive algorithm scale with N? The pre-smoothing and post-smoothing Jacobi or Gauss-Seidel iterations are the most time consuming parts of the calculation. Recall that a single Jacobi or Gauss-Seidel iteration scales like $\mathcal{O}(N^2)$. The operations must be carried out on the sequence of grids with

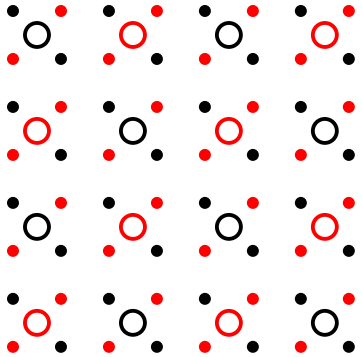
$$2^\ell \to 2^{\ell-1} \to 2^{\ell-2} \to \ldots \to 2^0 = 1$$

interior lattice points in each direction. The total number of operations is of order

$$N^2 \sum_{n=0}^{\ell} \frac{1}{2^{2n}} \leq N^2 \frac{1}{1 - \frac{1}{4}} \, .$$

Thus the multigrid $V-$cycle scales like $\mathcal{O}(N^2)$, i.e., linearly with the number of lattice points!

Restricting the Residual to a Coarser Lattice



The coarser lattice with spacing $H = 2h$ is constructed as shown. A simple algorithm for *restricting* the residual to the coarser lattice is to set its value to the average of the values on the four surrounding lattice points (cell-centered coarsening):

$$R_{I,J} = \frac{1}{4}\left[r_{i,j} + r_{i+1,j} + r_{i,j+1} + r_{i+1,j+1}\right] \ , \ i = 2(I-1) \ , \ j = 2(J-1) \ .$$

Prolongation of the Correction to the Finer Lattice

Having restricted the residual to the coarser lattice with spacing $H = 2h$, we need to solve the equation

$$\nabla^2 V = -R(x,y) \ ,$$

with the initial guess $V(x,y) = 0$. This is done by two-grid iteration

$$V = \texttt{twogrid}(\ell - 1, H, V, R) \ .$$

The output must now be interpolated or *prolongated* to the finer lattice. The simplest procedure is to copy the value of $V_{I,J}$ on the coarse lattice to the 4 neighboring cell points on the finer lattice:

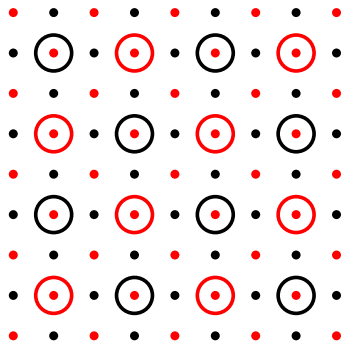
$$v_{i,j} = v_{i+1,j} = v_{i,j+1} = v_{i+1,j+1} = V_{I,J} \ , \quad i = 2(I-1) \ , \ j = 2(J-1) \ .$$

Improvements and More Complicated Multigrid Algorithms

The algorithm implemented above is the simplest multigrid scheme with a single *V-cycle*. Section 19.6 of *Numerical Recipes* discusses various ways of improving this algorithm:

- One can repeat the two-grid iteration more than once. If it is repeated *twice* in each multigrid level one obtains a *W-cycle* type of algorithm.

- The *Full Multigrid Algorithm* starts with the coarsest grid on which the equation can be solved exactly. It then proceeds to finer grids, performing one or more *V-cycles* at each level along the way. *Numerical Recipes* gives a program `mglin(u,n,ncycle)` which accepts the source function $-f$ in the first argument and implements the full multigrid algorithm with $\ell = \log_2(n-1)$ levels, performing `ncycle` V-cycles at each level, and returning the solution in the array parameter `u`. Note that this program assumes that the number of lattice points in each dimension N is odd, which leads to vertex centered coarsening:



This coarsening prescription doubles the lattice spacing *exactly*, unlike the *cell-centered* prescription for even N which slightly shifts the boundary points. However, it becomes a little more complicated to specify satisfactory restriction and prolongation operators because each coarse grid point now coincides with a fine grid point and is surrounded by 8 other fine grid points.

- More accurate algorithms for the restriction and prolongation operators can be used. A popular choice for the prolongation operator is to use *bilinear interpolation* in which the value of V at a coarse grid point is copied to 9 neighboring fine-grid points with the following weights:

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} .$$

For this to work, it is necessary that the restriction operator be the *adjoint* of the prolongation operator:

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} .$$