

Workshop #: Relaxation Methods for Boundary Value Problems

Poisson's Equation

Reading: Numerical Recipes, Ch 20.5

Consider the boundary value problem defined by the Poisson equation in the unit square

$$-\nabla^2 \phi = f(x, y), \quad \Omega = \{x, y\} \in [0, 1] \quad \phi = 0, \quad \{x, y\} \in \partial\Omega. \quad (1)$$

As a test problem for relaxation methods, consider the source term

$$f(x, y) = [2 + \pi^2(1 - y)y] \sin \pi x + [2 + \pi^2(1 - x)x] \sin \pi y, \quad (2)$$

which admits the exact solution

$$\phi(x, y) = y(1 - y) \sin \pi x + x(1 - x) \sin \pi y. \quad (3)$$

-
1. Write a program to calculate the solution via the Jacobi method:

$$u_{i,j}^{(k+1)} = \frac{1}{4} \left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)} + \Delta^2 f_{i,j} \right). \quad (4)$$

Evaluate the sum-of-squares error vs. the number of iterations.

2. Do the same for the Gauss-Seidel method:

$$u_{i,j}^{(k+1)} = \frac{1}{4} \left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} + \Delta^2 f_{i,j} \right). \quad (5)$$

3. Now try Successive Over-Relaxation (SOR):

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} - \omega \frac{\xi_{i,j}}{4}, \quad (6)$$

where the residual $\xi_{i,j} = u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} + \Delta^2 f_{i,j}$ and $1 < \omega < 2$. Compare a couple of suboptimal values of ω with the “optimal” value $\omega \simeq 2/(1 + \pi/J)$, where J is the number of grid points in each dimension.

4. Verify the number of iterations r , required to reduce the error by 10^{-p} , scales as expected for each method:

$$r \simeq \frac{1}{2} p J^2 \quad (\text{Jacobi}) \quad r \simeq \frac{1}{4} p J^2 \quad (\text{Gauss-Seidel}) \quad r \simeq \frac{1}{3} p J \quad (\text{SOR}) \quad (7)$$

5. [†] Improve the SOR method by implementing *odd-even ordering* on the mesh-points, and *Chebyshev acceleration* on the relaxation parameter:

$$\begin{aligned} \omega^{(0)} &= 1 \\ \omega^{(1/2)} &= 1/(1 - \rho_{\text{Jacobi}}^2/2) \\ \omega^{(n+1/2)} &= 1/(1 - \rho_{\text{Jacobi}}^2 \omega^{(n)}/4); \quad n = 1/2, 1, \dots \end{aligned} \quad (8)$$

6. [†] Direct matrix methods, although impractical for large problems, can be useful for small or medium sized problems. An example direct matrix code is provided in `directMatrixPoisson.cpp`, which makes use of the C++ linear algebra library `Eigen` to solve this system with sparse matrices and LU factorization. Compare how the speed and accuracy of the direct matrix method scales with J compared to a relaxation method.

[†]Optional