Programming Assignment 1: Multigrid for Poisson's Equation

Mech 511

Due Date: March 4

This programming problem examines some issues that arise in solving the two-dimensional Laplace equation using multigrid. Before embarking on the multigrid part of the program, you will want to be sure that you have a working point Gauss-Seidel over-relaxation scheme for a single mesh on the domain $(x,y) \in [0,1] \times [0,1]$ with zero boundary conditions. Unless otherwise specified, use a 64×64 mesh.

1. The first step towards a fully-operational multigrid code is writing and verifying the mesh-to-mesh transfer operators. Write code for the fine-to-coarse transfer operator and one of the coarse-to-fine transfer operators described in class. Test them by transferring the function $\sin(\pi x)\sin(\pi y)$ (evaluated at CV centers, naturally) from fine to coarse and back again. Plot the original function, the coarsened version, and the result of transferring back to the fine mesh. On a second plot, show the error in the transferred functions. For the simplest possible transfer operators, my errors are never larger in magnitude than about $3 \cdot 10^{-4}$ on the coarse mesh and about 0.025 after transferring back to the fine mesh. Using interpolation instead of injection for coarse-to-fine transfer improves the latter result, giving an error norm of around $2.5 \cdot 10^{-3}$.

2. Next, you need to verify that your source term on the coarse mesh is behaving as it should. To do this, use a two-level scheme. That is, for a single V-cycle, perform one smoothing pass on the finest mesh, 20 passes on a single coarse mesh, and one pass on the fine mesh. For this scheme, with initial data of $\sin(\pi x)\sin(\pi y)$ my code converges to an L_2-norm of the solution[1] of 10^{-9} in under 150 iterations with $\omega = 2/3$.

[1]The fully-converged solution is zero, remember.

3. Now you're ready to roll. Write a general V-cycle routine (with only two smoothing passes on the coarsest mesh) and test it by solving the same simple problem of part 2 on 1, 2, 3, 4 and 5 meshes. Plot convergence rate as measured by L_2-norm of the solution versus iteration for each case; use one plot for all cases. Also, tabulate the number of iterations to reach a convergence level of 10^{-9}.

4. *Comparison of relaxation parameter.*

   For the previous cases, you have used the (theoretically) best value of ω for multigrid performance. Using four mesh levels and the problem of part 2, compare convergence rates for the following values of ω: 0.5, 0.6, 0.666667, 0.8, 1.0, 1.25, and 1.5.[2] Finally, replace each single pass of smoothing in your code with two passes[3] using the values of ω that you found in the multigrid assignment.

   Tabulate the number of iterations required to reach a convergence level of 10^{-9}, remembering to double the number you get for the two-pass scheme. Why do too-large values of ω not work as well as lower values? Why does the optimum value for ω found in the problem set fail to give the best results here?

5. Finally, confirm that your code shows a convergence rate independent of the fine mesh size. To do this, solve the Poisson problem with zero boundary conditions and a forcing term of $x(1-x)y(1-y)$, starting in each case with an initial condition of $\sin(\pi x)\sin(\pi y)$. Use finest meshes of 32×32, 64×64, and 128×128, at least. (Feel free to go finer if you like!) In each case, your coarsest mesh should be 4×4. Use your choice of relaxation parameters. Tabulate the number of V-cycles required for convergence for each mesh size. Do your results meet expectations?

[2] Note that the last value is still well below the optimum value for a single mesh, which is about 1.8.

[3] Hint: Considering writing an intermediate-level smoothing routine that calls your Gauss-Seidel relaxation routine. This will make the two-pass code a trivial matter of adding a second call, instead of a copy-paste-modify exercise.