Math 228B
Homework 5
Due Friday, 3/22/19

1. Write a program to solve the discrete Poisson equation on the unit square using preconditioned conjugate gradient. Set up a test problem and compare the number of iterations and efficiency of using (i) no preconditioning, (ii) SSOR preconditioning, (iii) multigrid preconditioning. Run your tests for different grid sizes. How does the number of iterations scale with the number of unknowns as the grid is refined? For multigrid preconditioning, compare the efficiency of with multigrid as a standalone solver.

2. The file `cg_hw_matrix.mat` (or `cg_hw_matrix.bin` for those not using MATLAB) contains a 1000×1000 symmetric positive definite matrix.

   (a) Based on the condition number alone, according to the theoretical bound on the error how many iterations will be needed to insure a reduction in error by 10^{-6}?

   (b) Solve the system $A\vec{u} = \vec{b}$ for some different choices of $\vec{b}$. How many iterations did it take CG to converge?

   (c) Why are the results from parts (a) and (b) so different? Explain the difference based on how the CG algorithm works.

3. I provided code to give a matrix and right hand side for a discretized Poisson equation on a domain which is the intersection of the interior of the unit square and exterior of a circle centered at $(0.3, 0.4)$ with radius 0.15. The boundary conditions are zero on the square and 1 on the circle.

   Use your preconditioned conjugate gradient code to solve this problem. Explore the performance of no preconditioning and multigrid preconditioning for different grid sizes. Comment on your results. Note that the MG code is based on an MG solver for a different domain, and so it cannot be used as a solver for this problem. Is it an effective preconditioner?

SSOR preconditioning Symmetric SOR (SSOR) consists of one forward sweep of SOR followed by one backward sweep of SOR. For the discrete Poisson equation, one step of SSOR is

$$u_{i,j}^{k+1/2} = \frac{\omega}{4}(u_{i-1,j}^{k+1/2} + u_{i,j-1}^{k+1/2} + u_{i+1,j}^{k} + u_{i,j+1}^{k} - h^2 f_{i,j}) + (1-\omega)u_{i,j}^{k}$$
$$u_{i,j}^{k+1} = \frac{\omega}{4}(u_{i-1,j}^{k+1/2} + u_{i,j-1}^{k+1/2} + u_{i+1,j}^{k+1} + u_{i,j+1}^{k+1} - h^2 f_{i,j}) + (1-\omega)u_{i,j}^{k+1/2}.$$

It can be shown that one step of SSOR in matrix form is equivalent to

$$\frac{1}{\omega(2-\omega)}(D - \omega L)D^{-1}(D - \omega U)(\boldsymbol{u}^{k+1} - \boldsymbol{u}^{k}) = \boldsymbol{f},$$

where $A = D - L - U$.

For the constant coefficient problem, this suggests the preconditioner.

$$M = (D - \omega L)(D - \omega U).$$

Multigrid preconditioning To use MG as a preconditioner, the product $M^{-1}r$ is computed by applying one V-cycle with zero initial guess with right hand side r. If the smoother is symmetric and the number of pre and post smoothing steps are the same, this preconditioner is symmetric and definite and may be used with CG. Note that GSRB is **not** symmetric. You can use ω-Jacobi for the smoother or use a symmetric version of GS (at twice the cost). If you do not have working multigrid code, please let me know so that I can provide you with code for this portion of the assignment. You may also use a classmate's code, but please note that in your write-up.

Note: If you are interested, experiment with incomplete Cholesky factorization preconditioning. Incomplete Cholesky preconditioning requires that you form the matrix. Vary the amount of fill (in MATLAB use `cholinc` and vary the drop tolerance). Obviously, a factorization with more elements results in fewer iterations of CG, but it is more expensive to compute and to apply the preconditioner.