
Answer: See h4p1.m.


Answer:
From p. 114, the Richardson iteration matrix is $I - \omega A$. We follow the model of equation (13.65) and frequently use the fact that $A$ is symmetric, so $(Au, v) = (u, Av)$.

$$\|S(\omega)e\|_A^2 = (A(I - \omega A)e, (I - \omega A)e) = (Ae, e) - 2\omega(Ae, Ae) + \omega^2(A^2e, Ae) = \|e\|_A^2 - ((2\omega I - \omega^2 A)Ae, Ae) = \|e\|_A^2 - (D^{1/2}(2\omega I - \omega^2 A)D^{1/2} - 1/2 A)e, D^{-1/2}Ae) \leq \|e\|_A^2 - \lambda_{\min}(D^{1/2}(2\omega I - \omega^2 A)D^{1/2})\|Ae\|_2^2 \leq \lambda_{\min}(W)\|z\|$$

(In the last line we used the fact that $\|Wz\| \geq \lambda_{\min}(W)\|z\|$, just as Saad did in (13.65).)

Therefore the reduction factor is $\alpha = \lambda_{\min}(D^{1/2}(2\omega I - \omega^2 A)D^{1/2})$.

3. (15) Modify the multigrid program at http://www.cs.umd.edu/users/oleary/SCSCwebpage/cs_multigrid to use the Richardson iteration (Saad p. 114) instead of Gauss-Seidel as a smoother. Compare the performance of the two methods on the sample problems, using various values of $\alpha$.

Comments on the Answer:
Gauss-Seidel generally works better than Richardson on these problems and has the further advantage of not requiring a good guess at an unknown parameter. When implementing this, notice that computing $(I - \alpha A)u$ is much slower than computing $u - \alpha A(u)$, which is much slower than computing $u - \alpha (A^*u)$. Small details make big differences in implementation!

4. (5-35 points) Use GMRES to solve $Ax = b$ where $A$ is the matrix obtained from load west0479. Set the true solution to be the vector with every entry equal to 1. Use a restart parameter of 20 and a tolerance of $10^{-4}$. Experiment with various options for the preconditioner:

- (5) luinc changing the drop tolerance.
• (5) \texttt{luinc} using the modified version to preserve row sums.
• (5) \texttt{luinc} using a matrix reordering before factorization.
• (5) \texttt{luinc} using left, right, and two-sided preconditioning.
• (15) approximate inverse preconditioning.

Compare the performance of the methods with the performance of no preconditioner. Discuss.

Partial Answer: See h4p4.m.

• For ILU with a drop tolerance greater than about $10^{-6}$, the \texttt{U} matrix is singular, so the preconditioning fails for this problem.
• MILU works better on this problem than ILU for this problem (smaller time and smaller number of iterations).
• The \texttt{colamd} reordering reduces the number of nonzeros by a factor of 2 or so for this problem, and it is definitely worth using.
• If you constrain the sparsity structure of the approximate inverse preconditioner, then the columns can be computed by solving very small least squares problems. This takes some time but gives a very effective preconditioner on this problem. The sample code uses QR to solve the least squares problems; a Cholesky factorization of the normal equations matrix would be even faster, and since we are just computing a preconditioner, ill-conditioning in the least squares problem is not a concern.