AMATH 301 - Spring 2018

Homework #4

Due on Thursday, April 26, 2018

1. Consider the linear system $A_n\phi = \rho$, where A_n is an $n \times n$ matrix with 2's on the main diagonal, -1's directly above and below the main diagonal and 0's everywhere else. For instance, A_5 is

$$A_5 = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}.$$

This is a discretized version of Poisson's equation:

$$-\frac{d^2\phi}{dx^2} = \rho.$$

This equation appears very often in physical applications. We will discuss discretizations and differential equations, including the origin of this matrix A_n, later in the class.

Construct the matrix A_{80} in Matlab. In addition, make the 80×1 vector ρ such that the jth entry of ρ is defined according to the formula

$$\rho(j) = 2e^{-50\pi/81} \sin\left(\frac{50\pi j}{81}\right).$$

(a) The Jacobi iteration method for this problem can be written as $\phi_{k+1} = M\phi_k + \mathbf{c}$. (Note that ϕ_k means the kth guess for ϕ and is a vector. It does not mean the kth entry of ϕ.) Find the eigenvalue of M that has the largest absolute value (or magnitude for complex numbers) and save the absolute value (or magnitude) of this eigenvalue in `A1.dat`.

(b) Use Jacobi iteration to solve for ϕ. Your initial guess should be a 80×1 vector of all ones, and you should stop when the infinity norm of $\phi_{k+1} - \phi_k$ goes below a tolerance of 10^{-4}. (Recall that you can calculate the infinity norm of a vector `x` in Matlab using the command `norm(x,Inf)`.) Save the total number

of iterations in `A2.dat`. The initial guess does not count as an iteration, but every other guess does. Calculate the "true solution" ϕ using the backslash command, and find the error between your final guess and the true solution. Save the infinity norm of this error in `A3.dat`.

(c) The Gauss-Seidel iteration method for this problem can also be written as $\phi_{k+1} = M\phi_k + \mathbf{c}$. Note that these are not the same M and $\mathbf{c}$ as in part (a). Find the eigenvalue of M that has the largest absolute value (or magnitude for complex numbers) and save the absolute value (or magnitude) of this eigenvalue in `A4.dat`.

(d) Use Gauss-Seidel iteration to solve for ϕ. Your initial guess and stopping criterion should be the same as in part (b). Save the total number of iterations in `A5.dat`. Find the error between your final guess and the "true solution" ϕ and save the infinity norm of this error in `A6.dat`.

Things to think about: Is the matrix A_{80} strictly diagonally dominant? Does that matter? Based on the eigenvalues you found in parts (a) and (b), do you think this method is efficient? Which of the two methods would you expect to be faster and why? If you change n, how does it change the eigenvalues of M? What does this say about the speed of the method? Did you find a particularly accurate solution using either of the methods? In practice, would you use one of these methods to solve this problem?

2. Any matrix A can be decomposed into a diagonal, upper, and lower part:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix},$$

$$D = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix} \quad U = \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ 0 & 0 & \ddots & a_{2n} \\ \vdots & \vdots & \ddots & a_{n-1,n} \\ 0 & 0 & \dots & 0 \end{pmatrix} \quad L = \begin{pmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{pmatrix}$$

so that $A = D + U + L$. These matrices are useful for devising different matrix splitting methods. For instance, the Jacobi method uses $A = P + T$ where $P = D$ and $T = U + L$. The Gauss-Seidel method uses $P = D + L$ and $T = U$. Another common matrix splitting method is called successive over-relaxation (SOR). It is exactly the same as the splitting methods we have already seen, except we set

$$P = \frac{1}{\omega}D + L, \quad \text{and} \quad T = \frac{\omega - 1}{\omega}D + U,$$

where ω is a constant between 1 and 2. (Notice that if $\omega = 1$ then this is Gauss-Seidel.) We will use this method to solve the system $A_{80}\phi = \rho$ from problem 1.

(a) For any given ω, the SOR method can be written as $\phi_{k+1} = M\phi_k + \mathbf{c}$. For $\omega = 1.2$, find the eigenvalue of M that has the largest absolute value (or magnitude for complex numbers) and save the absolute value (or magnitude) of this eigenvalue in `A7.dat`.

(b) For every value of ω between $\omega = 1$ and $\omega = 1.99$ in increments of 0.01, find the eigenvalue of M that has the largest absolute value (or magnitude for complex numbers). Find the value of ω that corresponds to the smallest of these eigenvalues (in absolute value) and save the value of ω in `A8.dat`. (You may want to look up the `min` command in the Matlab help.)

(c) Use SOR to solve for ϕ with the optimal ω that you found in part (b). Your initial guess and stopping criterion should be the same as in problem 1(b) and 1(d). Save the total number of iterations in `A9.dat`. Find the error between your final guess and the "true solution" ϕ and save the infinity norm of this error in `A10.dat`.

Things to think about: Plot the eigenvalues from part (b) versus ω. What do you think would happen if you chose a slightly different value of ω from the one we used in part (c)? How does the eigenvalue you found for `A8.dat` compare to the ones from problem 1? What does this tell you about how fast the method will converge? Does this match with the number of steps the method took in part (c)? Was this method more or less accurate than the other two? Try plotting the result of each method on the same axes as the "true solution." Overall, do you think the gains in efficiency and accuracy are worth the time taken to find a good ω?