Homework 4

Problem 1: Poisson's Equation

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 example, 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

$$\frac{\mathrm{d}^2\phi(x)}{\mathrm{d}x^2} = \rho,$$

which 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_{100} in Matlab. (Look at the help file for the `diag` command; you can make this matrix in a few lines of code.) In addition, make the 100×1 vector ρ such that the jth entry of ρ is defined according to the formula

$$\rho(j) = 2\left(1 - \cos\left(\frac{56\pi}{101}\right)\right)\sin\left(\frac{56\pi j}{101}\right).$$

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

(b) Use Jacobi iteration to solve for ϕ. Your initial guess should be a 100×1 vector of all ones, and you should use a tolerance of 10^{-4}. That is, you should stop when

$$\|\phi_k - \phi_{k-1}\|_\infty < 10^{-4}.$$

(Remember, $\|\cdot\|_\infty$ denotes the infinity norm. You can find the infinity norm of a vector `x` in Matlab with 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, then find the error between your final guess and the true solution. Save the infinity norm of this error (i.e., $\|\phi_k - \phi\|_\infty$) in `A3.dat`.

(c) The Gauss-Seidel iteration method for this problem can also be written as $\phi_k = M\phi_{k-1} + \mathbf{c}$. (Note that these are not the same M and $\mathbf{c}$ as in part (a).) Find the largest (in magnitude) eigenvalue of M and save the 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_{100} 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 would you expect to be faster and why? As you change n, how does it change the speed of the two methods? Did you find a particularly accurate solution using either of the methods? Are you surprised by the accuracy?

Problem 2: Successive Over-Relaxation

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

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

$$L = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ a_{21} & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{pmatrix},$$

so that $A = D+U+L$. These matrices are useful for devising different matrix splitting methods. For instance, in the Jacobi method we let $A = P + T$ where $P = D$ and

$T = U + L$ and in the Gauss-Seidel method we set $P = D + L$ and $U = T$. 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

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

where ω is a constant between 1 and 2. (Notice that if $\omega = 1$ then this is exactly the same as Gauss-Seidel.)

We will use this method to solve the system $A_{100}\phi = \rho$ from problem 1.

(a) For any given ω, the SOR method can be written as $\phi_k = M\phi_{k-1} + \mathbf{c}$. For $\omega = 1.5$, find the largest (in magnitude) eigenvalue of M and save the 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 largest (in magnitude) eigenvalue of M. Find the value of ω that corresponds to the smallest of these eigenvalues and save it in `A8.dat`.

(c) Use SOR to solve for ϕ with the optimal ω 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 in A8 compare to the ones from problem 1? What does this tell you about the speed of the method? 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? Overall, do you think the gains in efficiency/accuracy are worth the time taken to find a good ω?