AMATH 301 – Autumn 2019
Homework 4

Due: 7:00pm, October 30, 2019.

Instructions for submitting:

- Scorelator problems are submitted as a MATLAB script (`.m` file). You should **NOT** upload your `.dat` files. The `.dat` files should be created by your script. You have 5 attempts on Scorelator.
- Writeup problems are submitted to Gradescope as a single `.pdf` file that contains text and plots. Put the problems in order and label each writeup problem. When you submit, **you must indicate which problem is which on Gradescope**. **All code you used for this part of the assignment should be included either at the end of the problem or at the end of your .pdf file.**

Scorelator problems

1. **Random walks**

   This part is background. If you are interested, read on. Don't spend much time trying to figure it out if you are struggling.

   A *random walk* is a mathematical idea from the field of *probability*. Random walks have been used to model polymer configurations, metallurgy, trapping of electrons in solar cells, the motion of microorganisms on surfaces, Brownian motion, population dynamics, etc.

   The main idea for a random walk is that an object can be at any one of $n+2$ equally spaced points $x_0, \ldots, x_{n+1}$ and can only move between adjacent points and does so with equal probability. In other words, if the object is at x_i it can move to either x_{i-1} or x_{i+1}.

   An important question that can come up is: what is the probability that an object at location x_i will reach the left endpoint (x_0) before reaching the right endpoint (x_{n+1})? We label the probability that the object at x_i will reach the left endpoint before the right endpoint p_i. From the above rules, $p_0 = 1$ and $p_{n+1} = 0$. Our goal in this problem is to find

$$\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}.$$

Using the above rules, we can find

$$p_i = \frac{1}{2}p_{i-1} + \frac{1}{2}p_{i+1}, \qquad \text{for each } i = 1, 2, \ldots, n.$$

We do so by solving

$$\boldsymbol{A}_n \mathbf{p} = \mathbf{b}_n$$

where

$$\mathbf{b}_n = \begin{pmatrix} 1/2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

If the object can be in $n + 2 = 7$ possible locations, we have

$$\boldsymbol{A}_5 = \begin{pmatrix} 1 & -1/2 & 0 & 0 & 0 \\ -1/2 & 1 & -1/2 & 0 & 0 \\ 0 & -1/2 & 1 & -1/2 & 0 \\ 0 & 0 & -1/2 & 1 & -1/2 \\ 0 & 0 & 0 & -1/2 & 1 \end{pmatrix}, \qquad \mathbf{b}_5 = \begin{pmatrix} 1/2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

In other words, A_n is the matrix with 1 on the main diagonal, and $-1/2$ on the upper and lower diagonal.

(a) Construct the matrix $\boldsymbol{A}_{90}$ in Matlab. The `diag` command should be helpful here.

(b) Solve the linear system $\boldsymbol{A}_{90}\mathbf{p} = \mathbf{b}_{90}$ using the backslash command. Save the solution in `A1.dat`. This will serve as the "exact solution" for the error calculations below.

(c) Recall that the formula for the Jacobi method is

$$\mathbf{p}_{k+1} = -\mathbf{D}^{-1}\mathbf{T}\mathbf{p}_k + \mathbf{D}^{-1}\mathbf{b},$$

where $\mathbf{D}$ is a diagonal matrix and $\boldsymbol{A}_{90} = \mathbf{D} + \mathbf{T}$. This can be written more compactly as

$$\mathbf{p}_{k+1} = \mathbf{M}\mathbf{p}_k + \mathbf{c},$$

where $\mathbf{M} = -\mathbf{D}^{-1}\mathbf{T}$ and $\mathbf{c} = \mathbf{D}^{-1}\mathbf{b}$. Find the eigenvalue of $\mathbf{M}$ that has the largest absolute value, and save the absolute value of this eigenvalue in `A2.dat`.

(d) Use the Jacobi method to solve for $\mathbf{p}$. Your initial guess should be a 90×1 vector of all ones, and you should use a tolerance of 10^{-5}. Your tolerance check should use the infinity-norm, *i.e.,* `norm(xkplus1 - xk, Inf) < tol`. Calculate the error by subtracting the final iteration of the Jacobi method, $\mathbf{p}_{\text{Jacobi}}$, from the "exact solution" obtained in part (b), $\mathbf{p}_{\text{exact}}$, and then taking the 2-norm of the difference:

$$\text{error} = \|\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{Jacobi}}\|_2.$$

Create a 1×2 row vector where the first entry is the number of iterations it took for the method to converge and the second entry is the error:

$$\begin{pmatrix}\text{number of iterations} & \text{error}\end{pmatrix}.$$

Save the vector in `A3.dat`.

(e) The formula for the Gauss-Seidel method can also be written as $\mathbf{p}_{k+1} = \mathbf{M}\mathbf{p}_k + \mathbf{c}$, but for a different matrix $\mathbf{M}$ and vector $\mathbf{c}$. Find the eigenvalue of $\mathbf{M}$ that has the largest absolute value, and save the absolute value of this eigenvalue in `A4.dat`.

(f) Use the Gauss-Seidel method to solve for $\mathbf{p}$. Your initial guess and stopping criterion should be the same as in part (d). Create a 1×2 row vector with the number of iterations and the error, and save it in `A5.dat`.

2. Any matrix $\boldsymbol{A}$ can be decomposed into a diagonal, upper, and lower part:

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

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

so that $\boldsymbol{A} = \mathbf{D} + \mathbf{U} + \mathbf{L}$. These matrices are useful for devising different matrix splitting methods in which the matrix $\boldsymbol{A}$ is split into the sum of two matrices

$$\boldsymbol{A} = \mathbf{P} + \mathbf{T}.$$

For instance, the Jacobi method uses $\mathbf{P} = \mathbf{D}$ and $\mathbf{T} = \mathbf{U} + \mathbf{L}$. The Gauss-Seidel method uses $\mathbf{P} = \mathbf{D} + \mathbf{L}$ and $\mathbf{T} = \mathbf{U}$. 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

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

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 $\boldsymbol{A}_{90}\mathbf{P} = \mathbf{b}_{90}$ from problem 1.

For any given ω, the SOR method can be written as $\boldsymbol{x}_{k+1} = \mathbf{M}\boldsymbol{x}_k + \mathbf{c}$. **You should find M just as we have done for Jacobi and Gauss-Seidel.** The convergence of SOR is determined by the eigenvalues of M. SOR converges fastest at what is called the *optimal value of* ω. The optimal value of ω is the value of ω such that the largest eigenvalue of the matrix M is smaller than the largest eigenvalue for all other choices of ω (in absolute value). Our goal is to find ω.

(a) Split $\boldsymbol{A} = \mathbf{P} + \mathbf{T}$ and find the matrix $\mathbf{M}$ for the SOR method.

(b) For every value of ω between (and including) $\omega = 1$ and $\omega = 1.99$ in increments of 0.01, find the eigenvalue of $\mathbf{M}$ that has the largest absolute value. Save the largest eigenvalue for each $\mathbf{M}$ in a row vector and save that vector in `A6.dat`.

(c) Find the optimal value of ω and save it in `A7.dat`. (You may want to look up the `min` command.)

(d) Use the SOR method to solve $\boldsymbol{A}_{90}\mathbf{p} = \mathbf{b}_{90}$ for $\mathbf{p}$ with the optimal ω that you found in part (c). Your initial guess and stopping criterion should be the same as in problem 1. Create a 1×2 row vector with the number of iterations and the error, and save it in `A8.dat`.

3. Is the matrix $\boldsymbol{A}_{90}$ in Problem 1 strictly diagonally dominant? Does this determine for sure whether or not the Jacobi and Gauss-Seidel methods will converge? Save your answer in `A9.dat`.

A The matrix $\boldsymbol{A}_{90}$ **is not** strictly diagonally dominant which implies that the Gauss-Seidel method will converge but does not guarantee convergence for the Jacobi method.

B The matrix $\boldsymbol{A}_{90}$ **is** strictly diagonally dominant which implies that the Jacobi and Gauss-Seidel methods will not converge.

C The matrix $\boldsymbol{A}_{90}$ **is not** strictly diagonally dominant which implies that the Jacobi and Gauss-Seidel methods will converge.

D The matrix $\boldsymbol{A}_{90}$ **is** strictly diagonally dominant which implies that the Gauss-Seidel method will converge but does not guarantee convergence for the Jacobi method.

E The matrix $\boldsymbol{A}_{90}$ **is** strictly diagonally dominant which implies that the Jacobi and Gauss-Seidel methods will converge.

F The matrix $\boldsymbol{A}_{90}$ **is not** strictly diagonally dominant so the two methods may converge but it is not guaranteed.

G The matrix $\boldsymbol{A}_{90}$ **is not** strictly diagonally dominant which implies that the Jacobi and Gauss-Seidel methods will not converge.

H The matrix $\boldsymbol{A}_{90}$ **is** strictly diagonally dominant which implies that the Jacobi method will converge but does not guarantee convergence for the Gauss-Seidel method.

I The matrix $\boldsymbol{A}_{90}$ **is not** strictly diagonally dominant which implies that the Jacobi method will converge but does not guarantee convergence for the Gauss-Seidel method.

4. Assume that we want to solve $F\mathbf{x} = \mathbf{b}$. Assuming that an iterative method would converge, which of the following matrices would be best suited for solving using an iterative method? Save your answer in `A10.dat`. **Note:** The $\cdots$ means that the pattern continues.

$$\mathbf{A}\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{1000} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{1001} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{1002} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{1000} & \frac{1}{1001} & \frac{1}{1002} & \cdots & \frac{1}{1999} \end{pmatrix} \qquad \mathbf{B}\begin{pmatrix} .1 & 0 & 0 & 0 \\ -.03 & 0.9 & 0 & 0 \\ 8 & -8 & 1.2 & 0 \\ \pi & -2 & -2.5 & 9 \end{pmatrix}$$

$$\mathbf{C}\begin{pmatrix} 0 & \sin(1) & 0 & \sin(3) & 0 & \sin(5) & 0 & \sin(7) \\ \sin(1) & 0 & \sin(3) & 0 & \sin(5) & 0 & \sin(7) & 0 \\ 0 & \sin(3) & 0 & \sin(5) & 0 & \sin(7) & 0 & \sin(9) \\ \sin(3) & 0 & \sin(5) & 0 & \sin(7) & 0 & \sin(9) & 0 \\ 0 & \sin(5) & 0 & \sin(7) & 0 & \sin(9) & 0 & \sin(11) \\ \sin(5) & 0 & \sin(7) & 0 & \sin(9) & 0 & \sin(11) & 0 \\ 0 & \sin(7) & 0 & \sin(9) & 0 & \sin(11) & 0 & \sin(13) \\ \sin(7) & 0 & \sin(9) & 0 & \sin(11) & 0 & \sin(13) & 0 \end{pmatrix}$$

$$\mathbf{D}\begin{pmatrix} 4 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 100 & 4 & 2 & 0 & \cdots & 0 & 0 \\ 0 & 90 & 4 & 3 & \ddots & 0 & 0 \\ 0 & 0 & 98 & 4 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & 4 & 9000 \\ 0 & 0 & 0 & 0 & \cdots & -8899 & 4 \end{pmatrix} \qquad \mathbf{E}\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & -0.03 \end{pmatrix}$$

Gradescope problems

1. The following parts need to be completed. **The only parts you need to include in your writeup are: the figure in part (b), the table in part (g), and the comments in part (h).**

   (a) Solve $\boldsymbol{A}_{90}\mathbf{p} = \mathbf{b}_{90}$ again using the Jacobi method, the Gauss-Seidel method, and the SOR method with the optimal ω. Use an error tolerance of 10^{-6} for each method and a random vector as your initial guess (using `randn` in MATLAB). For each method use the "exact solution," calculated in Scorelator Problem 1 part (b), to calculate the absolute error for each of the probabilities $p_1, \ldots, p_{90}$. This should be a vector of length 90 for each method. In other words, calculate

   $$\begin{aligned} \text{error}_{\text{Jacobi}} &= |\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{Jacobi}}|, \\ \text{error}_{\text{GS}} &= |\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{GaussSeidel}}|, \\ \text{error}_{\text{SOR}} &= |\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{SOR}}|, \end{aligned}$$

   elementwise, where $\mathbf{p}_{\text{exact}}$ is the exact solution found in Scorelator Problem 1(b), and $\mathbf{p}_{\text{Jacobi}}$, $\mathbf{p}_{\text{GS}}$, and $\mathbf{p}_{\text{SOR}}$ are the approximate solutions obtained by the Jacobi, Gauss-Seidel, and SOR iterations respectively.

   (b) Create a figure with three plots using the `subplot` command to split the figure into an upper, middle, and lower axis. Your figure should have a title (the `sgtitle` command is useful here) and axes labels for each of the subplots. The top window should show the following.

   - The true solution in black with small dots connected by solid lines.
   - The solution obtained by Jacobi iteration plotted as red circles without lines in between.
   - The solution obtained by the Gauss-Seidel method plotted as green diamonds without lines in between.
   - The solution obtained by the SOR method, with the optimal ω, plotted as blue squares without lines in between.

   All of the above should be plotted against the vector

   $$\begin{pmatrix} 1 & 2 & \cdots & 90 \end{pmatrix}.$$

   In the middle window, plot the Jacobi error as red circles connected by red lines, the Gauss-Seidel error as green diamonds connected by green lines, and the SOR error as blue squares connected by blue lines. Again the error should be plotted against the above vector.

   In the bottom window, create the same plot as the middle window except using `semilogy` instead of `plot`.

   Your axes labels, title, and linewdith in your plot should all be an appropriate size to make them readable.

   (c) Calculate the largest eigenvalue (in absolute value) of $\mathbf{M}$ for the Jacobi method, the Gauss-Seidel method, and the SOR method with the optimal ω.

(d) Record the number of iterations required for each of the methods to reach the desired error tolerance.

(e) Calculate $\lambda_{\text{J}}^{\text{Jiter}}$, $\lambda_{\text{GS}}^{\text{GSIter}}$, and $\lambda_{\text{SOR}}^{\text{SORIter}}$ where λ_{J}, λ_{GS}, and λ_{SOR} are the eigenvalues for Jacobi, Gauss-Seidel, and SOR you found in part (c) above and where JIter, GSIter, and SORIter are the number of iterations required for Jacobi, Gauss-Seidel, and SOR found in part (d). In other words, raise the eigenvalue with the largest magnitude to the power of the number of iterations.

(f) Calculate the three errors

$$\begin{aligned}(\text{Total error})_{\text{Jacobi}} &= \|\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{Jacobi}}\|_2 \\ (\text{Total error})_{\text{GS}} &= \|\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{GaussSeidel}}\|_2 \\ (\text{Total error})_{\text{SOR}} &= \|\mathbf{p}_{\text{exact}} - \mathbf{p}_{\text{SOR}}\|_2\end{aligned}$$

Reminder: $\|\boldsymbol{x}_{\text{exact}} - \boldsymbol{x}_n\|_2$ means use the 2-norm, or `norm()` in MATLAB.

(g) Fill in the following table:

	Largest eigenvalue of $\mathbf{M}$	Num. Iterations	λ^{iter}	Total Error
Jacobi				
Gauss-Seidel				
SOR				

The first column should have your answers from part (c), the second should have your answers from part (d), the third should have your answers from part (e), and the fourth should have your answers from part (f).

(h) Comment on the error plot and the table above. Your comments should include a discussion of:

i. For what values of n in p_n is the error the largest? You should comment on how this relates to the original problem.

ii. Do you see any major differences in the error for the three different methods?

iii. Which method converges the fastest (in the least number of iterations)? You should comment on how this relates to the eigenvalues.

iv. How does the error in the table compare with the λ^{iter} column? What similarities do you see?

Finally, which method do you think is the best for this problem?