Hardeep Bassi
09/12/2021

Math 231 Homework #1

1 (a) (i) From lecture, we know that the Jacobi iterative update is given by:

↳ $x_i^{(k)} = \frac{1}{a_{ii}} \{ b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k-1)} \}$ for $i \neq j$ and $a_{ii} \neq 0$.

Using this, we can write the component-wise recursion as:

$$\boxed{\begin{aligned} &\bullet\ x_1^{k} = \left(1 - x_2^{(k-1)} + 2x_3^{(k-1)}\right)/2 \\ &\bullet\ x_2^{k} = \left(2 - x_1^{(k-1)} - x_3^{(k-1)}\right)/1 \\ &\bullet\ x_3^{k} = \left(3 - 3x_1^{(k-1)} - 2x_2^{(k-1)}\right)/1 \end{aligned}}$$

(ii) From lecture, we derived the Jacobi matrix recursion formula as:

↳ $Dx^{k} = b - (L+U)x^{(k-1)}$ for $A = (L+D+U)$.

This implies that, by the invertibility of D:

↳ $x^{k} = D^{-1}b - D^{-1}(L+U)x^{(k-1)}$ (*)

Using this, A_1 decomposes as:

$$A_1 = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 3 & 2 & 0 \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & 1 & -2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{U}$$

By (*), we get: $x^{k} = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & 1 \\ 3 & 2 & 0 \end{bmatrix} x^{(k-1)}$

$$\Rightarrow \boxed{x^{(k)} = \begin{bmatrix} 1/2 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 0 & -1/2 & 1 \\ -1 & 0 & -1 \\ -3 & -2 & 0 \end{bmatrix} x^{(k-1)}}$$

(iii) We know that the Jacobi iterative method will converge if the spectral radius of matrix form recurrence relation is less than 1.

By (ii), the matrix $B_J = \begin{bmatrix} 0 & -1/2 & 1 \\ -1 & 0 & -1 \\ 3 & -2 & 0 \end{bmatrix}$ is the iteration matrix.

↳ Using Matlab, we see $\rho(B_J) = \boxed{0.9208} < 1$

∴ $\boxed{\text{Jacobi will converge for } A_1}$

→

① (b)(i) From lecture, we know that the Gauss-Seidel update method is given by:

↳ $x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i-1}^{n} a_{ij} \cdot x_j^{(k-1)}\right)$

Using this, the component-wise recursion becomes:

$$\boxed{\begin{aligned} &\bullet\ x_1^{(k)} = (1 - x_2^{(k-1)} - x_3^{(k-1)})/2 \\ &\bullet\ x_2^{(k)} = (2 - x_1^{k} - x_3^{(k-1)})/1 \\ &\bullet\ x_3^{(k)} = (3 - 3x_1^{k} - 2x_2^{k})/1 \end{aligned}}$$

(ii) From lecture, we derived the Gauss-Seidel matrix update as:

↳ $x^k = (D+L)^{-1} b - (D+L)^{-1} U x^{(k-1)}$ for $A = (L+D+U)$.

Using the same decomposition from (a) and Matlab to carry out the matrix multiplications, we get:

↳ $x^k = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & -2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

$\Rightarrow \boxed{x^k = \begin{bmatrix} 1/2 \\ 3/2 \\ -3/2 \end{bmatrix} + \begin{bmatrix} 0 & -1/2 & 1 \\ 0 & 1/2 & -2 \\ 0 & 1/2 & 1 \end{bmatrix} x^{(k-1)}}$

(iii) Using the recurrence iteration matrix $B_{GS} = \begin{bmatrix} 0 & -1/2 & 1 \\ 0 & 1/2 & -2 \\ 0 & 1/2 & 1 \end{bmatrix}$, we can obtain $\rho(B_{GS}) = \boxed{1.2247} > 1$.

↳ $\boxed{\text{Hence by our convergence theorem, Gauss-Seidel will not converge for matrix } A.}$

1) c) (i) Using the Jacobi iterative method on page 11 for A_2, we get:

$$\boxed{\begin{aligned} \bullet\ x_1^k &= \left(1 - x_2^{(k-1)} - 3x_3^{(k-1)}\right)/2 \\ \bullet\ x_2^k &= \left(2 - x_1^{(k-1)} - x_3^{(k-1)}\right)/2 \\ \bullet\ x_3^k &= \left(3 - x_1^{(k-1)} - x_2^{(k-1)}\right)/3 \end{aligned}}$$

(ii) As before, the matrix-form recursion for the Jacobi method is given by:

↳ $x^k = D^{-1}b - D^{-1}(L+U)x^{(k-1)}$

Using this, A_2 decomposes as:

↳ $A_2 = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & 1 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{U}$

$\Rightarrow x^k = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}\begin{bmatrix} 0 & 1 & 3 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} x^{(k-1)}$

$$\boxed{\Rightarrow x^k = \begin{bmatrix} 1/2 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & -1/2 & -3/2 \\ -1/2 & 0 & -1/2 \\ -1/3 & -1/3 & 0 \end{bmatrix} x^{(k-1)}}$$

(iii) By (ii), we see $B_J = \begin{bmatrix} 0 & -1/2 & -3/2 \\ -1/2 & 0 & -1/2 \\ -1/3 & -1/3 & 0 \end{bmatrix}$. Using Matlab, we obtain:

↳ $\boxed{\rho(B_J) = 1.1039 > 1}$

∴ $\boxed{\text{Jacobi will not converge for } A_2}$ since B_J has a spectral radius greater than 1.

(d) (i) Using the Gauss-Seidel iteration method from page (2), we get:

$$\boxed{\begin{aligned} &\bullet\ x_1^{(k)} = \left(1 - x_2^{(k-1)} - 3x_3^{(k-1)}\right)/2 \\ &\bullet\ x_2^{(k)} = \left(2 - x_1^{k} - x_3^{(k-1)}\right)/2 \\ &\bullet\ x_3^{(k)} = \left(3 - x_1^{k} - x_2^{k}\right)/3 \end{aligned}}$$

(ii) From page (2), we have the matrix-form recursion as:

$\hookrightarrow x^k = (D+L)^{-1} b - (D+L)^{-1} U x^{(k-1)}$ for $A = (L+D+U)$.

Using the decomposition of A_2 on the previous page & Matlab, we get:

$$\hookrightarrow x^k = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x^{(k-1)}$$

$$\Rightarrow \boxed{x^k = \begin{bmatrix} 0.5 \\ 0.75 \\ 0.5833 \end{bmatrix} + \begin{bmatrix} 0 & -0.5 & -1.5 \\ 0 & 0.25 & 0.25 \\ 0 & 0.0833 & 0.4166 \end{bmatrix} x^{(k-1)}}$$

(iii) Using the recurrence iteration matrix $B_{GS} = \begin{bmatrix} 0 & -0.5 & -1.5 \\ 0 & 0.25 & 0.25 \\ 0 & 0.0833 & 0.4166 \end{bmatrix}$,

we observe its spectral radius as: (Matlab)

$\hookrightarrow \boxed{\rho(B_{GS}) = 0.506021}$

$\hookrightarrow$ Hence by the convergence theorem, Gauss-Seidel converges for A_2.

② (a) Given $Ax=b$ as below, we can recast into form $x^k = B_J x^{(k-1)} + C_J$:

$$\Rightarrow \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

For the Jacobi method, we know that by lecture:

$$\Rightarrow x^{(k)} = D^{-1} b - D^{-1}(L+U)x^{(k-1)} \text{ for } A = (L+D+U) \quad (*)$$

$$\Rightarrow L = \begin{bmatrix} 0 & 0 \\ c & 0 \end{bmatrix}, \; D = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}, \; U = \begin{bmatrix} 0 & b \\ 0 & 0 \end{bmatrix}$$

Using this decomposition of A, alongside $(*)$, we get:

$$\hookrightarrow x^{(k)} = \begin{bmatrix} 1/a & 0 \\ 0 & 1/d \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} 1/a & 0 \\ 0 & 1/d \end{bmatrix} \begin{bmatrix} 0 & b \\ c & 0 \end{bmatrix} x^{(k-1)}$$

$$= \begin{bmatrix} u/a \\ v/d \end{bmatrix} + \begin{bmatrix} 0 & -b/a \\ -\frac{c}{d} & 0 \end{bmatrix} x^{(k-1)}$$

$$\Rightarrow \boxed{C_J = \begin{bmatrix} u/a \\ v/d \end{bmatrix} \quad B_J = \begin{bmatrix} 0 & -b/a \\ -c/d & 0 \end{bmatrix}}$$

(b) • $\|B_J\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$. This, in turn, means the absolute maximum sum across the columns of B_J. Using B_J from part (a), we get:

$$\Rightarrow \boxed{\max\left\{ \left|-\frac{c}{d}\right|, \left|-\frac{b}{a}\right| \right\}}$$

• $\|B_J\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$. This translates to absolute maximum row sum.

$$\Rightarrow \boxed{\max\left\{ \left|-\frac{b}{a}\right|, \left|-\frac{c}{d}\right| \right\}}$$

• $\|B_J\|_2 = \max_{1 \le i \le n} \{ \sqrt{\lambda_i} : \lambda_i \text{ an eigenvalue of } B_J^T B_J \}$

$$\Rightarrow \begin{bmatrix} 0 & -c/d \\ -b/a & 0 \end{bmatrix} \begin{bmatrix} 0 & -b/a \\ -\frac{c}{d} & 0 \end{bmatrix} = \begin{bmatrix} c^2/d^2 & 0 \\ 0 & b^2/a^2 \end{bmatrix} = B_J^T B_J$$

∴ The eigenvalues of $B_J^T B_J$ are along the diagonal since $B_J^T B_J$ is upper triangular. Hence $\lambda_i = \{\frac{c^2}{d^2}, \frac{b^2}{a^2}\}$, therefore

$$\Rightarrow \|B_J\|_2 = \boxed{\max\left\{ \pm\frac{c}{d}, \pm\frac{b}{a} \right\}}$$ using the definition above

• $\rho(B_J) = \max_{1 \le i \le n} \{ |\lambda_i| : \lambda_i \text{ an eigenvalue of } B_J \}$

$$\Rightarrow \det(B_J - \lambda \mathbb{I}) = \begin{bmatrix} -\lambda & -b/a \\ -c/d & -\lambda \end{bmatrix} = \lambda^2 - \frac{bc}{ad} = 0$$

$$\Rightarrow \lambda = \boxed{\pm\sqrt{\frac{bc}{ad}}}$$

$$\Rightarrow \boxed{\rho(B_J) = \max\left\{ \left| \pm\sqrt{\frac{bc}{ad}} \right| \right\}}$$

② (c) By the calculations in (b), we see that they can be sorted as:

↳ $\rho(B_J) \le \|B_J\|_2 = \|B_J\|_1 = \|B_J\|_\infty$

This is because all of $\|B_J\|_2$, $\|B_J\|_1$, and $\|B_J\|_\infty$ are defined to be the maximum of either $\frac{b}{a}$ or $\frac{c}{d}$. Observing $\rho(B_J)$:

↳ $\rho(B_J) = \left|\sqrt{\frac{bd}{ac}}\right| = \left|\sqrt{\frac{b}{d}} \cdot \sqrt{\frac{d}{c}}\right|$. Without a loss of generality, assume $\left|\frac{b}{a}\right| \ge \left|\frac{c}{d}\right|$

$\Rightarrow \rho(B_J) = \left|\sqrt{\frac{b}{a}} \cdot \sqrt{\frac{c}{d}}\right| \le \left|\sqrt{\frac{b}{a}} \cdot \sqrt{\frac{b}{a}}\right| = \left|\frac{b}{a}\right| = \|B_J\|_1 = \|B_J\|_2 = \|B_J\|_\infty$

because $\left|\frac{b}{a}\right| \ge \left|\frac{c}{d}\right|$. Similarly, the argument applies if $\frac{b}{a} < \frac{c}{d}$ or if $\frac{b}{a} = \frac{c}{d}$.

Hence, $\boxed{\rho(B_J) \le \|B_J\|_2 = \|B_J\|_1 = \|B_J\|_\infty}$

(d) By our convergence theorem, we must have that $\rho(B_J) < 1$ for Jacobi convergence.

$\Rightarrow \rho(B_J) = \left|\pm\sqrt{\frac{bc}{ad}}\right| < 1$

$\Rightarrow \boxed{ad > bc}$ allows the quotient to be less than 1 and consequently guarantee convergence.

(e) Let $A = \begin{bmatrix} 100 & 0.1 \\ 0.1 & 200 \end{bmatrix}$. To check our condition in (d), we see: $\Rightarrow \boxed{\sqrt{\frac{(0.1)^2}{20000}} < 1}$ ✓.

Hence, Jacobi should converge. To verify:

$\Rightarrow B_J = \begin{bmatrix} 0 & -\frac{0.1}{100} \\ -\frac{0.1}{200} & 0 \end{bmatrix}$

∴ Using Matlab, we see the maximum absolute eigenvalue is

$\Rightarrow \lambda = 7.07106 \times 10^{-4} < 1$ ✓

↳ $\boxed{\text{Hence, our condition in (d) holds for diagonally dominant } A.}$ ✓

MATH 231 Homework 1 MATLAB

1. Write the following MATLAB functions for the Jacobi and Gauss-Seidel methods to solve a general n x n system `A*x=b`:

   - `function [final sol,sols] = Jacobi(A,b,x0,niter)`
   - `function [final sol,sols] = GaussSeidel(A,b,x0,niter)`

   Here, `final_sol` is the iterative solution after `niter` iterations starting with initial guess `x0` and `sols` is the sequence of iterative solutions $\{x^{(0)}, x^{(1)}, ..., x^{(niter)}\}$ (n x $(niter + 1)$ matrix).

SOLUTION:

```
JACOBI METHOD
function [final_sol, sols] = Jacobi(A,b,x0,niter)
x = x0;
n = size(A);
xnew = zeros(n(1), 1);
sols = [x];
for k = 1:niter
    for i =1:n
        xnew(i) = (b(i) - A(i, 1:i-1) * x(1:i-1) - A(i, i+1:n) * x(i+1:n))/
            (A(i,i));
    end
    x = xnew;
    sols = [sols x];
end
final_sol = sols(:,end);
```

```
GAUSS-SEIDEL
function [final_sol, sols] = GaussSeidel(A,b,x0,niter)
x = x0;
sols = [x];
n = size(A);
for k = 1:niter
    for i=1:n
        x(i) = (b(i) - A(i, 1:i-1) * x(1:i-1) - A(i, i+1:n) * x(i+1:n))/
            (A(i,i));
    end
    sols = [sols x];
end
final_sol = sols(:,end);
```

Using the pseudocode provided in both lectures, the Jacobi method and the Gauss-Seidel method were implemented in MATLAB using the code above.

In the Jacobi method we update our initial guess vector `x0` using the recurrence relation derived in lecture, being sure to skip over the diagonal entries, hence splitting our indexing on the `A` and `x` into two separate parts for the subtractions. We then update the `x` vector to be the result of going all the rows of `A` according to this recurrence relation, stored in `xnew`, and append the result to the `sols` array for `niter` iterations.

Similarly, in the Gauss-Seidel method, the psuedocode from lecture is implemented. Instead of allocating a new vector `xnew`, the Gauss-Seidel method is able to iteratively update the entries of the `x` vector according to it's recurrence relation. Because of how our relation is defined, we can access previous indices of `x` if we have already computed them, and use them in our update rule. For the entries that we have yet to compute, we can access our previous iteration's value at that index of `x`. Hence, the first indexing on `A` and `x` represents the use of values on indices that we have already calculated, whereas the second indexing on `A` and `x` represents indices in `x` that we still must calculate, hence resorting us to access the previous iteration of `x` on these indices. We conclude by dividing by the diagonal entry, marking the final indexing on `A`. We also store the results in the `sols` array and continue this across the remaining iterations.

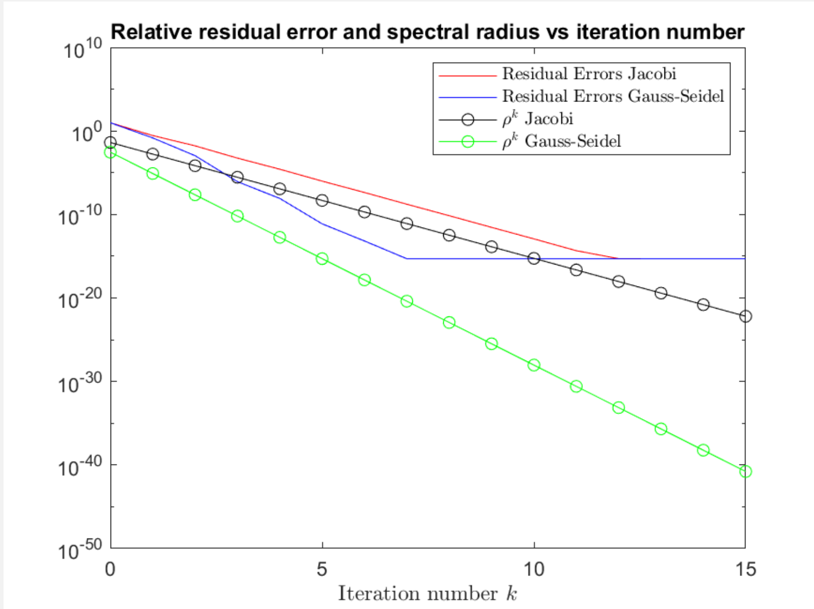
2. For each method, validate your implementation by using a 6 x 6 system (convergent case).
   - Plot relative residual errors versus iteration number k (using semi-log plot).
   - Compute the spectral radius ρ of the iteration matrix and plot ρ^k versus k (compare the slopes).

SOLUTION:

Both methods are known to work on diagonally dominant matrices, hence if we set

$$\texttt{A} = \begin{bmatrix} 100 & 2 & 4 & 1 & 2 & 3 \\ 1 & 200 & 1 & 1 & 2 & 3 \\ 4 & 4 & 300 & 0 & 1 & 2 \\ 7 & 1 & 2 & 400 & 3 & 1 \\ 5 & 7 & 2 & 1 & 500 & 1 \\ 0 & 0 & 0 & 0 & 0 & 600 \end{bmatrix}, \texttt{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}, \texttt{x0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

we can expect convergence for both methods. As discussed in class, the initial guess vector `x0` and the `b` vector do not impact the convergence of these methods.
The parameters used for both methods were the above `A`, `b`, `x0`, as well as `niter` $= 15$. Using the script found in the appendix, we generate the graph below.



For the iteration matrix $B_{GS} = -(D+L)^{-1}U$, we see that the spectral radius $\rho(B_{GS}) = 0.00282882$ and for the iteration matrix $B_J = -D^{-1}(L+U)$, we see that the spectral radius $\rho(B_J) = 0.0410156$, where $A = L + D + U$ is used to find L, D, U. Since both spectral radii are less than 1, we expect to see convergence. As seen in the graph, the residual errors for both methods quickly tend to 0 (subject to computer arithmetic). Since the slope of the ρ^k line for the Gauss-Seidel method decreases much faster than that of the Jacobi method, we see Gauss-Seidel converging to the minimum computable error in less iterations than that of the Jacobi method (Iteration 7 for Gauss-Seidel vs Iteration 12 for Jacobi).

3. For each method, observe the divergent case.
   - Plot relative residual errors versus iteration number k (using semi-log plot).
   - Compute the spectral radius ρ of the iteration matrix and plot ρ^k versus k (compare the slopes).

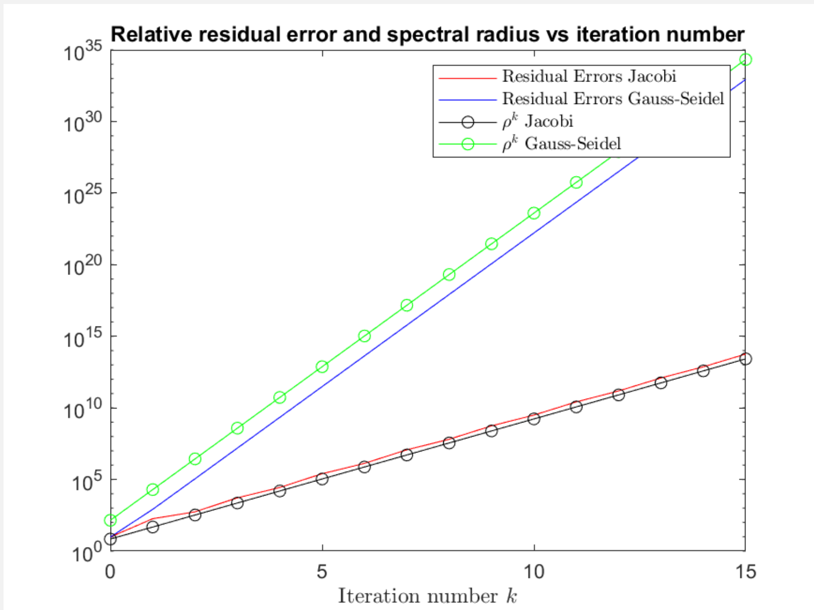
SOLUTION:

To demonstrate the divergent case, we would need to construct a matrix such that the spectral radius of the iteration matrix B is greater than 1. Using the `randn` MATLAB function, we generate a random matrix shown below. We use the same set parameters as we did in the convergent case for `b`, `x0`, and `niter`, and define `A` below. Using the script found in the appendix, we generate the graph below.

$$\texttt{A} = \begin{bmatrix} -1.0667 & -0.084539 & 0.23235 & 2.2294 & 0.42272 & 0.32706 \\ 0.93373 & 1.6039 & 0.42639 & 0.33756 & -1.6702 & 1.0826 \\ 0.35032 & 0.098348 & -0.37281 & 1.0001 & 0.47163 & 1.0061 \\ -0.029006 & 0.041374 & -0.23645 & -1.6642 & -1.2128 & -0.65091 \\ 0.18245 & -0.73417 & 2.0237 & -0.59003 & 0.06619 & 0.25706 \\ -1.5651 & -0.030814 & -2.2584 & -0.27806 & 0.65236 & -0.94438 \end{bmatrix},$$

$$\texttt{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}, \texttt{x0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

As seen in the graph below, we have the spectral radius computed by MATLAB of both iteration matrices $\rho(B_J) = 6.89978$ and $\rho(B_{GS}) = 139.837$, being greater than 1, where B_J and B_{GS} are defined as they were in part 2. We see that as k increases, both spectral radii explode, as well as the residual errors for the Jacobi and Gauss-Seidel methods. Since the Gauss-Seidel method has a much higher spectral radius for it's iteration matrix, we see that the slope of it's ρ^k line is much larger than that of the Jacobi method, and it's residual error as a result explodes much faster. Both methods share divergence however; getting further away from the true solution at an exponential rate.



APPENDIX

```
clear all; close all; clc;


%A = [-1.0667  -0.084539    0.23235     2.2294     0.42272     0.32706;
%     0.93373      1.6039    0.42639    0.33756    -1.6702      1.0826;
%     0.35032    0.098348   -0.37281     1.0001     0.47163      1.0061;
%   -0.029006    0.041374   -0.23645    -1.6642    -1.2128    -0.65091;
%     0.18245    -0.73417     2.0237   -0.59003     0.06619     0.25706;
%     -1.5651   -0.030814    -2.2584   -0.27806     0.65236    -0.94438];
A = [100 2 4 1 2 3; 1 200 1 1 2 3; 4 4 300 0 1 2; 7 1 2 400 3 1; 5 7 2 1 500
    1; 0 0 0 0 0 600];
b=[1;2;3;4;5;6];
x0=[0;0;0;0;0;0];
niter=15;

% Jacobi
[final_solJ,solsJ] = Jacobi(A,b,x0,niter);

% Gauss-Seidel
[final_solGS, solsGS] = GaussSeidel(A,b,x0,niter);

%iteration matrix
D = diag(diag(A));
L = tril(A, -1);
U = triu(A, 1);

%Jacobi
it_matrixJ = -inv(D) * (L+U);

%Gauss-Seidel
it_matrixGS = -inv(D+L) * U;

% residual errors
errsJ = [];
errsGS = [];
for i=1:niter+1
    errJ=norm(A*solsJ(:,i)-b);
    errsJ=[errsJ errJ];
    errGS=norm(A*solsGS(:,i)-b);
    errsGS=[errsGS errGS];
end


spec_radiusJ = max(abs(eig(it_matrixJ)));
spec_radiusGS = max(abs(eig(it_matrixGS)));
radiiJ = [];
radiiGS = [];
for j=1:niter+1
    radiusJ = spec_radiusJ ^ j;
    radiiJ = [radiiJ radiusJ];
    radiusGS = spec_radiusGS ^ j;
    radiiGS = [radiiGS radiusGS];
end
```

```
% semi-log plot
semilogy(0:niter,errsJ,'r')
hold on
semilogy(0:niter, errsGS, 'b')
hold off


%spectral radius plot
hold on
plot(0:niter, radiiJ, 'black-o')
hold off
hold on
plot(0:niter, radiiGS, 'g-o')
hold off

xlabel("Iteration number $k$", 'Interpreter', 'latex')
title('Relative residual error and spectral radius vs iteration number')
legend('Residual Errors Jacobi','Residual Errors Gauss-Seidel', '$\rho ^ k$
    Jacobi', '$\rho ^k$ Gauss-Seidel', 'Interpreter', 'latex')
%saveas(gcf, 'convergent.png')
%change to divergent.png and uncomment the correct matrix and comment out the
    diagonally dominant one
```