Nicholas Garrett

Professor Zhu

MATH 4441

11/10/2021

Exam 2

1.

$$\begin{bmatrix} x(x_0) & log(x_0) & cos(x_0) \\ x(x_1) & log(x_1) & cos(x_1) \\ x(x_2) & log(x_2) & cos(x_2) \\ ... & ... & ... \\ x(x_n) & log(x_n) & cos(x_n) \end{bmatrix} * \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ ... \\ y_n \end{bmatrix}$$

2.

See code

3.

For Jacobi: For the matrix where $a_{11}, a_{22} \neq 0$, the way to determine if it converges or diverges is through the special radius of the B matrix $\rho(B) : B = D^{-1}(L+U)x_n$.

D $= \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix} \Rightarrow D^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 \\ 0 & \frac{1}{a_{22}} \end{bmatrix}$

U $= \begin{bmatrix} 0 & a_{12} \\ 0 & 0 \end{bmatrix}$ and L $= \begin{bmatrix} 0 & 0 \\ a_{21} & 0 \end{bmatrix} \Rightarrow L + U = \begin{bmatrix} 0 & a_{12} \\ a_{21} & 0 \end{bmatrix}$

$\Rightarrow B = \begin{bmatrix} \frac{1}{a_{11}} & 0 \\ 0 & \frac{1}{a_{22}} \end{bmatrix} * \begin{bmatrix} 0 & a_{12} \\ a_{21} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 \end{bmatrix}$

$\lambda(B) = [-\frac{\sqrt{a_{12}}\sqrt{a_{22}}}{\sqrt{a_{11}}\sqrt{a_{21}}}, \frac{\sqrt{a_{12}}\sqrt{a_{22}}}{\sqrt{a_{11}}\sqrt{a_{21}}}]$

$\rho(B) = max(| \pm \frac{\sqrt{a_{12}}\sqrt{a_{22}}}{\sqrt{a_{11}}\sqrt{a_{21}}}|)$

For Gauss-Seidel: For the matrix where $a_{11}, a_{22} \neq 0$, the way to determine if it converges or diverges is through the special radius of the B matrix $\rho(B) : B = (D-L)^{-1}(U)$.

$(D-L) = \begin{bmatrix} a_{11} & 0 \\ -a_{21} & a_{22} \end{bmatrix} \Rightarrow (D-L)^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 \\ \frac{a_{21}}{a_{11}a_{22}} & \frac{1}{a_{22}} \end{bmatrix}$

$(D-L)^{-1} * U = \begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} \\ 0 & \frac{a_{12}*a_{21}}{a_{11}*a_{22}} \end{bmatrix}$

$\lambda(B) = [0, \frac{a_{12}a_{21}}{a_{11}a_{22}}]$

$\rho(B) = max(|0, \frac{a_{12}a_{21}}{a_{11}a_{22}}|)$

The speed of convergence really depends on the values for $a_{11}, a_{12}, a_{21}, a_{22}$ quite a bit. For the algorithms to converge, the matrix A must be row diagonally-dominant, which would imply that for Jacobi or Gauss-Seidel iterative methods to work, $a_{11} > a12$ and $a_{21} < a_{22}$. Therefore, $max(| \pm \frac{\sqrt{a_{12}}\sqrt{a_{22}}}{\sqrt{a_{11}}\sqrt{a_{21}}}|) > max(| \pm \frac{\sqrt{a_{12}}\sqrt{a_{22}}}{\sqrt{a_{11}}\sqrt{a_{21}}}|)$, as the latter holds the maximum of 0 and $\frac{a_{12}a_{21}}{a_{11}a_{22}}$, which is smaller than the maximum of $max(| \pm \frac{\sqrt{a_{12}}\sqrt{a_{22}}}{\sqrt{a_{11}}\sqrt{a_{21}}}|)$ due to the relative sizes of a_{11} and a_{22} to a_{12} and a_{21}. Therefore, the Gauss-Seidel iterative method converges faster.

4.

4.a:

$\rho(B) : B = D^{-1}(L + U)x_n$:

$$D^{-1} = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \end{bmatrix}$$

$$D^{-1}(L+U) = \begin{bmatrix} 0 & -\frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 1 \\ -\frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

$\lambda(B) = [\frac{1}{2}]$

$\rho(B) = \frac{1}{2}$

4.b.

See code.

The code ran for 58 iterations before MATLAB finally rounded to the exact values of 1, 2, and -1 for x. However, it became accurate to 5 decimals (the default display truncation) after 17 iterations.

4.c. $\rho(B) : B = (D - L)^{-1}(U)$:

$$(D-L) = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \end{bmatrix} \Rightarrow (D-L)^{-1} = \begin{bmatrix} 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ -1/2 & -1/4 & 1/2 \end{bmatrix}$$

$$(D-L)^{-1}(U) = \begin{bmatrix} 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{3}{2} \\ 0 & \frac{1}{2} & -1 \end{bmatrix}$$

$\lambda(B) = [0]$

$\rho(B) = 0$

4.d.
See Code

For my code, it took 27 iterations to get the ∞-norm $< 1 * 10^{-6}$

5.

5.a

eigenvalues: 0, 1, 3
I have a function in the MATLAB file, $Gauss Seidel_r elaxation$, which solves for the eigenvalues.

eigenvector for $\lambda = 0 : (1, 1, 1)^T$

$$Ax = \lambda x \Rightarrow \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} * x = \ lambda * x = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

5.b
see code.

ϵ	iterations
$\epsilon = 1$:	2
$\epsilon = 10^{-1}$:	183
$\epsilon = 10^{-2}$:	1908
$\epsilon = 10^{-3}$:	19164
$\epsilon = 10^{-4}$:	191718
$\epsilon = 0$:	(it is still running)

It is interesting to see how quickly an efficient algorithm can become inefficient. I did not realize how quickly the SOR algorithm becomes inefficient with ϵ of small values, though I suppose from looking at the iterative equation it should not be unexpected, with ϵ being multiplied across the iterative algorithm.