1. You are given this linear system of equations $Ax = b$ as follows

$$2x_1 - x_2 + x_3 = -1, \;\; 2x_1 + 2x_2 + 2x_3 = 4, \;\; -x_1 - x_2 + 2x_3 = -5.$$

(a) Applying the Gaussian elimination (Slide 19 in Module 05), compute the matrix inverse A^{-1} then find $x^* = A^{-1}b$.

(b) Compute the mapping matrices T for the Jacobi and the Gauss-Seidel methods and show that $\rho(T_{jacobi}) \approx 1.12$ and that $\rho(T_{gauss}) = 0.5$.

(c) Confirm the findings of part (b) via showcasing convergence/divergence of these two methods.

Problem 1 Solution:

(a) Compute A^{-1} via Gaussian elimination: Rewrite the system of linear equations as an augmented matrix, $[A|I]$. Let $R_n - R_m$ refer to the subtraction of row m from row n. Then the Gaussian elimination may be written:

$$\left[\begin{array}{ccc|ccc} 2 & -1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 1 & 0 \\ -1 & -1 & 2 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - R_1} \left[\begin{array}{ccc|ccc} 2 & -1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 1 & -1 & 1 & 0 \\ -1 & -1 & 2 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + \frac{1}{2}R_1}$$

$$\left[\begin{array}{ccc|ccc} 2 & -1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 1 & -1 & 1 & 0 \\ 0 & -3/2 & 5/2 & 1/2 & 0 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + \frac{1}{2}R_2} \left[\begin{array}{ccc|ccc} 2 & -1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 1 & -1 & 1 & 0 \\ 0 & 0 & 3 & 0 & 1/2 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - \frac{1}{3}R_3}$$

$$\left[\begin{array}{ccc|ccc} 2 & -1 & 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & -1 & 5/6 & -1/3 \\ 0 & 0 & 3 & 0 & 1/2 & 1 \end{array}\right] \xrightarrow{R_1 \leftarrow R_1 - \frac{1}{3}R_3} \left[\begin{array}{ccc|ccc} 2 & -1 & 0 & 1 & -1/6 & -1/3 \\ 0 & 3 & 0 & -1 & 5/6 & -1/3 \\ 0 & 0 & 3 & 0 & 1/2 & 1 \end{array}\right]$$

$$\xrightarrow{R_1 \leftarrow R_1 + \frac{1}{3}R_2} \left[\begin{array}{ccc|ccc} 2 & 0 & 0 & 2/3 & 1/9 & -4/9 \\ 0 & 3 & 0 & -1 & 5/6 & -1/3 \\ 0 & 0 & 3 & 0 & 1/2 & 1 \end{array}\right] \xrightarrow{\text{normalize}}$$

$$\implies A^{-1} = \begin{bmatrix} 1/3 & 1/18 & -2/9 \\ -1/3 & 5/18 & -1/9 \\ 0 & 1/6 & 1/3 \end{bmatrix}$$

Use A^{-1} to solve for x^*:

$$x^* = A^{-1}b = \begin{bmatrix} 1/3 & 1/18 & -2/9 \\ -1/3 & 5/18 & -1/9 \\ 0 & 1/6 & 1/3 \end{bmatrix} \begin{bmatrix} -1 \\ 4 \\ -5 \end{bmatrix} = \begin{bmatrix} -1/3 + 4/18 + 10/9 \\ 1/3 + 20/18 + 5/9 \\ 0 + 4/6 - 5/3 \end{bmatrix}$$

$$\implies x^* = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

(b) For the Jacobi method, $A = M - N$, where $M = D$ and $N = E + F$. Compute the mapping matrix T_{Jacobi}:

$$D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad E = \begin{bmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{bmatrix}$$

$$x^{(k)} = D^{-1}(E + F)\, x^{(k-1)} + D^{-1} b$$

$$\begin{aligned} \implies T_{Jacobi} &= D^{-1}(E + F) \\ &= \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 \\ -2 & 0 & -2 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & -1/2 \\ -1 & 0 & -1 \\ 1/2 & 1/2 & 0 \end{bmatrix} \end{aligned}$$

Compute $\rho\left(T_{Jacobi}\right) = \max\left|\text{eig}\left(T_{Jacobi}\right)\right|$:

$$\begin{aligned} \det\left(T_{Jacobi} - \lambda I_3\right) &= 0 \\ \begin{vmatrix} -\lambda & 1/2 & -1/2 \\ -1 & -\lambda & -1 \\ 1/2 & 1/2 & -\lambda \end{vmatrix} &= 0 \\ -\lambda\left(\lambda^2 + \frac{1}{2}\right) - \frac{1}{2}\left(\lambda + \frac{1}{2}\right) - \frac{1}{2}\left(-\frac{1}{2} + \frac{\lambda}{2}\right) &= 0 \\ -\lambda^3 - \frac{1}{2}\lambda - \frac{1}{2}\lambda - \frac{1}{4} + \frac{1}{4} - \frac{1}{4}\lambda &= 0 \\ -\lambda^3 - \frac{5}{4}\lambda &= 0 \\ -\lambda\left(\lambda^2 + \frac{5}{4}\right) &= 0 \end{aligned}$$

$$\implies \lambda = 0, \pm i\sqrt{5/4} \implies \rho\left(T_{Jacobi}\right) \approx 1.12$$

(c) For the Gauss-Seidel method, $A = M - N$, where $M = D - E$ and $N = F$. Compute the mapping matrix T_{Gauss}:

$$D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad E = \begin{bmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ -1 & -1 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{bmatrix}$$

$$x^{(k)} = (D - E)^{-1} F x^{(k-1)} + (D - E)^{-1} b$$

Compute $(D - E)^{-1}$:

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 & 1 & 0 \\ 1 & 1 & 2 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - R_1} \left[\begin{array}{ccc|ccc} 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & -1 & 1 & 0 \\ 1 & 1 & 2 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - \frac{1}{2}R_1}$$

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & -1 & 1 & 0 \\ 0 & 1 & 2 & -1/2 & 0 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - \frac{1}{2}R_2}$$

$$
\begin{aligned}
\implies T_{Gauss} &= (D-E)^{-1} F \\
&= \begin{bmatrix} 2 & 0 & 0 \\ 2 & 2 & 0 \\ 1 & 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1/2 & 0 & 0 \\ -1/2 & 1/2 & 0 \\ 0 & -1/4 & 1/2 \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1/2 & -1/2 \\ 0 & -1/2 & -1/2 \\ 0 & 0 & 1/2 \end{bmatrix}
\end{aligned}
$$

Compute $\rho\left(T_{Gauss}\right) = \max|\text{eig}\left(T_{Gauss}\right)|$:

$$
\begin{aligned}
\det\left(T_{Gauss} - \lambda I_3\right) &= 0 \\
\begin{vmatrix} -\lambda & 1/2 & -1/2 \\ 0 & -(1/2+\lambda) & -1/2 \\ 0 & 0 & 1/2-\lambda \end{vmatrix} &= 0 \\
-\lambda\left(\left(-\frac{1}{2}-\lambda\right)\left(\frac{1}{2}-\lambda\right)-0\right) - \frac{1}{2}(0-0) - \frac{1}{2}(0-0) &= 0 \\
-\lambda\left(-\frac{1}{4}+\frac{1}{2}\lambda-\frac{1}{2}\lambda+\lambda^2\right) &= 0 \\
-\lambda\left(\lambda^2-\frac{1}{4}\right) &= 0
\end{aligned}
$$

$$
\implies \lambda = 0, \pm\frac{1}{2} \implies \rho\left(T_{\text{Gauss}}\right) = 0.5
$$

2. Solve

$$x_1 + x_2 - x_3 = -3, \;\; 6x_1 + 2x_2 + 2x_3 = 2, \;\; -3x_1 + 4x_2 + x_3 = 1$$

using with Naive Gauss, Gaussian elimination with. partial pivoting, and Gauss-Jordan without partial pivoting:

Problem 2 Solution: Solve using 3 methods:

$$\begin{cases} x_1 + x_2 - x_3 = -3 \\ 6x_1 + 2x_2 + 2x_3 = 2 \\ -3x_1 + 4x_2 + x_3 = 1 \end{cases}$$

(a) Naive Gauss:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 6 & 2 & 2 & 2 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - 6R_1} \left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + 3R_1}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ 0 & 7 & -2 & -8 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + \frac{7}{4}R_2} \left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ 0 & 0 & 12 & 27 \end{array}\right]$$

$$\begin{cases} 12x_3 = 27 & \implies x_3 = 2.25 \\ 4x_2 + 8x_3 = 20 & \implies x_2 = -0.5 \\ x_1 + x_2 - x_3 = -3 & \implies x_1 = -0.25 \end{cases}$$

(b) Gaussian elimination with partial pivoting:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 6 & 2 & 2 & 2 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{\text{swap } R_1 \text{ \& } R_2} \left[\begin{array}{ccc|c} 6 & 2 & 2 & 2 \\ 1 & 1 & -1 & -3 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - \frac{1}{6}R_1}$$

$$\left[\begin{array}{ccc|c} 6 & 2 & 2 & 2 \\ 0 & 2/3 & -4/3 & -10/3 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + \frac{1}{2}R_1} \left[\begin{array}{ccc|c} 6 & 2 & 2 & 2 \\ 0 & 2/3 & -4/3 & -10/3 \\ 0 & 5 & 2 & 2 \end{array}\right] \xrightarrow{\text{swap } R_2 \text{ \& } R_3}$$

$$\left[\begin{array}{ccc|c} 6 & 2 & 2 & 2 \\ 0 & 5 & 2 & 2 \\ 0 & 2/3 & -4/3 & -10/3 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - \frac{2}{15}R_2} \left[\begin{array}{ccc|c} 6 & 2 & 2 & 2 \\ 0 & 5 & 2 & 2 \\ 0 & 0 & -8/5 & -18/5 \end{array}\right]$$

$$\begin{cases} -\frac{8}{5}x_3 = -\frac{18}{5} & \implies x_3 = 2.25 \\ 5x_2 + 2x_3 = 2 & \implies x_2 = -0.5 \\ 6x_1 + 2x_2 + 2x_3 = 2 & \implies x_1 = -0.25 \end{cases}$$

(c) Gauss-Jordan without partial pivoting:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 6 & 2 & 2 & 2 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - 6R_1} \left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ -3 & 4 & 1 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + 3R_1}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ 0 & 7 & -2 & -8 \end{array}\right] \xrightarrow{R_2 \leftarrow -\frac{1}{4}R_2} \left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & 1 & -2 & -5 \\ 0 & 7 & -2 & -8 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - 7R_2}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & 1 & -2 & -5 \\ 0 & 0 & 12 & 27 \end{array}\right] \xrightarrow{R_3 \leftarrow \frac{1}{12}R_3} \left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & 1 & -2 & -5 \\ 0 & 0 & 1 & 27/12 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 + 2R_3}$$

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & -3 \\ 0 & 1 & 0 & -1/2 \\ 0 & 0 & 1 & 27/12 \end{array}\right] \xrightarrow{R_1 \leftarrow R_1 + R_3} \left[\begin{array}{ccc|c} 1 & 1 & 0 & -3/4 \\ 0 & 1 & 0 & -1/2 \\ 0 & 0 & 1 & 27/12 \end{array}\right] \xrightarrow{R_1 \leftarrow R_1 - R_2}$$

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & -1/4 \\ 0 & 1 & 0 & -1/2 \\ 0 & 0 & 1 & 27/12 \end{array}\right]$$

$$\begin{aligned} &\implies x_1 = -1/4 \\ &\implies x_2 = -1/2 \\ &\implies x_3 = 27/12 \end{aligned}$$

3. Solve the following system of equations $Ax = b$ via the LU decomposition without pivoting:

$$15x_1 + 7x_2 - 4x_3 = -51, \quad 4x_1 - 4x_2 + 9x_3 = 62, \quad 12x_1 - x_2 + 3x_3 = 8$$

then determine the matrix inverse after you complete the reduced row echelon form and confirm that $A^{-1}A = I_3$.

Problem 3 Solution:

(a) Solve using LU decomposition:

$$\begin{aligned} 15x_1 + 7x_2 - 4x_3 &= -51 \\ 4x_1 - 4x_2 + 9x_3 &= 62 \\ 12x_1 - x_2 + 3x_3 &= 8 \end{aligned}$$

Compute L & U via reduced row echelon form:

$$\begin{bmatrix} 15 & 7 & -4 \\ 4 & -4 & 9 \\ 12 & -1 & 3 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 - \frac{4}{15}R_1} \begin{bmatrix} 15 & 7 & -4 \\ 4/15 & -88/15 & 151/15 \\ 12 & -1 & 3 \end{bmatrix} \xrightarrow{R_3 \leftarrow R_3 - \frac{12}{15}R_1}$$

$$\begin{bmatrix} 15 & 7 & -4 \\ 4/15 & -88/15 & 151/15 \\ 12/15 & -33/5 & 31/5 \end{bmatrix} \xrightarrow{R_3 \leftarrow R_3 - \frac{33}{5}\cdot\frac{15}{88}R_2} \begin{bmatrix} 15 & 7 & -4 \\ 4/15 & -88/15 & 151/15 \\ 12/15 & 9/8 & -41/8 \end{bmatrix}$$

$$\implies L = \begin{bmatrix} 1 & 0 & 0 \\ 4/15 & 1 & 0 \\ 12/15 & 9/8 & 1 \end{bmatrix} \quad \& \quad U = \begin{bmatrix} 15 & 7 & -4 \\ 0 & -88/15 & 151/15 \\ 0 & 0 & -41/8 \end{bmatrix}$$

we now have: $Ax = b \implies LUx = b$, where $Ly = b$ if $Ux = y$

Compute y:

$$Ly = b \implies \begin{bmatrix} 1 & 0 & 0 \\ 4/15 & 1 & 0 \\ 12/15 & 9/8 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -51 \\ 62 \\ 8 \end{bmatrix}$$

$$\begin{cases} y_1 = -51 & \implies y_1 = -51 \\ \frac{4}{15}y_1 + y_2 = 62 & \implies y_2 = 378/5 \\ \frac{12}{15}y_1 + \frac{9}{8}y_2 + y_3 = 8 & \implies y_1 = -145/4 \end{cases}$$

Compute x:

$$Ux = y \implies \begin{bmatrix} 15 & 7 & -4 \\ 0 & -88/15 & 151/15 \\ 0 & 0 & -41/8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -51 \\ 378/5 \\ -145/4 \end{bmatrix}$$

$$\begin{cases} -\frac{41}{8}x_3 = -\frac{145}{4} & \implies x_3 = 290/41 \\ -\frac{88}{15}x_2 + \frac{151}{15}x_3 = \frac{378}{5} & \implies x_2 = -338/451 \\ 15x_1 + 7x_2 - 4x_3 = -51 & \implies x_1 = -525/451 \end{cases}$$

(b) Determine the inverse of the coefficients matrix:

$$\left[\begin{array}{ccc|ccc} 15 & 7 & -4 & 1 & 0 & 0 \\ 4 & -4 & 9 & 0 & 1 & 0 \\ 12 & -1 & 3 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 - \frac{4}{15}R_1}$$

$$\left[\begin{array}{ccc|ccc} 15 & 7 & -4 & 1 & 0 & 0 \\ 0 & -88/15 & 151/15 & -4/15 & 1 & 0 \\ 12 & -1 & 3 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - \frac{12}{15}R_1}$$

$$\left[\begin{array}{ccc|ccc} 15 & 7 & -4 & 1 & 0 & 0 \\ 0 & -88/15 & 151/15 & -4/15 & 1 & 0 \\ 0 & -33/5 & 31/5 & -4/5 & 0 & 1 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - \frac{33}{5}\cdot\frac{15}{88}R_2}$$

$$\left[\begin{array}{ccc|ccc} 15 & 7 & -4 & 1 & 0 & 0 \\ 0 & -88/15 & 151/15 & -4/15 & 1 & 0 \\ 0 & 0 & -41/8 & -1/2 & -9/8 & 1 \end{array}\right] \xrightarrow{R_2 \leftarrow R_2 + \frac{151}{15}\cdot\frac{8}{41}R_3}$$

$$\left[\begin{array}{ccc|ccc} 15 & 7 & -4 & 1 & 0 & 0 \\ 0 & -88/15 & 0 & -256/205 & -248/205 & 1208/615 \\ 0 & 0 & -41/8 & -1/2 & -9/8 & 1 \end{array}\right] \xrightarrow{R_1 \leftarrow R_1 - 4\cdot\frac{8}{41}R_3}$$

$$\left[\begin{array}{ccc|ccc} 15 & 7 & 0 & 57/41 & 36/41 & -32/41 \\ 0 & -88/15 & 0 & -256/205 & -248/205 & 1208/615 \\ 0 & 0 & -41/8 & -1/2 & -9/8 & 1 \end{array}\right] \xrightarrow{R_1 \leftarrow R_1 + 7\cdot\frac{15}{88}R_2}$$

$$\left[\begin{array}{ccc|ccc} 15 & 0 & 0 & -45/451 & -255/451 & 705/451 \\ 0 & -88/15 & 0 & -256/205 & -248/205 & 1208/615 \\ 0 & 0 & -41/8 & -1/2 & -9/8 & 1 \end{array}\right] \xrightarrow{\text{normalize each row}}$$

$$\implies A^{-1} = \begin{bmatrix} -3/451 & -17/451 & 47/451 \\ 96/451 & 93/451 & -151/451 \\ 4/41 & 9/41 & -8/41 \end{bmatrix}$$

(c) Confirm $A^{-1}A = I_3$:

Multiplying $A^{-1}A$ does indeed result in I_3. Check using Matlab or Python.

4. Linear algebraic equations can arise in the solution of differential equations. For example, the following differential equation results from a steady-state mass balance for a chemical in a one-dimensional canal (e.g., concentrations of chlorine in pipes of a water distribution network in a city):

$$0 = D\frac{d^2c}{dx^2} - U\frac{dc}{dx} - kc$$

where $c(x)$ is the concentration as a function of x which is the distance. Parameters D, U, k define the diffusion coefficient, fluid velocity, and a first-order decay rate, respectively. Note that typically the concentrations are a function of space (i.e., location in the pipe) and time, but this model forgoes the time-element and focuses on studying concentrations as a function of space only.

Anyway, convert this differential equation to an equivalent system of simultaneous algebraic equations via using the central divided difference for the derivative (and second derivative), that we learned in previous homework assignments.

Solve these equations from $x = 0$ to $x = 10$ with $\Delta x = 2$, and develop a plot of concentration versus distance.The parameters given are: $D = 2.5, U = 0.75, k = 0.15$, with initial conditions $c(0) = 75$ and boundary conditions $c(10) = 25$. Solve the problem again when $\Delta x = 1$ and $\Delta x = 0.1$. What do you notice?

Plot the graphs of $c(x)$ versus x for different Δx.

Problem 4 Solution:

$$0 = D\frac{d^2c}{dx^2} - U\frac{dc}{dx} - kc \qquad \begin{array}{ll} D = 2.5 & c\,(0) = 75 \\ U = 0.75 & c\,(10) = 25 \\ k = 0.15 & \Delta x = 2 \end{array}$$

Using the definitions of the first and second central divided differences:

$$f'\,(x_i) \approx \frac{f\,(x_{i+1}) - f\,(x_{i-1})}{2\,(\Delta x)} \qquad f''\,(x_i) \approx \frac{f\,(x_{i+1}) - 2f\,(x_i) + f\,(x_{i-1})}{(\Delta x)^2}$$

$$\implies 0 = D \cdot \frac{c_{i+1} - 2c_i + c_{i-1}}{(\Delta x)^2} - U \cdot \frac{c_{i+1} - c_{i-1}}{2\,(\Delta x)} - kc_i$$

$$0 = \left(\frac{D}{(\Delta x)^2} - \frac{U}{2\Delta x}\right)(c_{i+1}) - \left(\frac{2D}{(\Delta x)^2} + k\right)(c_i) + \left(\frac{D}{(\Delta x)^2} + \frac{U}{2\Delta x}\right)(c_{i-1})$$

For $\Delta x = 2$:

$$0 = \frac{7}{16}c_{i+1} - \frac{7}{5}c_i + \frac{13}{16}c_{i-1}, \text{ for } i = 1, 2, 3, 4$$

$$\begin{cases} \frac{7}{16}c_2 - \frac{7}{5}c_1 + \frac{13}{16}c_0 = 0 \\ \frac{7}{16}c_3 - \frac{7}{5}c_2 + \frac{13}{16}c_1 = 0 \\ \frac{7}{16}c_4 - \frac{7}{5}c_3 + \frac{13}{16}c_2 = 0 \\ \frac{7}{16}c_5 - \frac{7}{5}c_4 + \frac{13}{16}c_3 = 0 \end{cases}$$

Plugging in initial conditions:

$$\begin{cases} \frac{7}{16}c_2 - \frac{7}{5}c_1 + \frac{13}{16}\,(75) = 0 \\ \frac{7}{16}c_3 - \frac{7}{5}c_2 + \frac{13}{16}c_1 = 0 \\ \frac{7}{16}c_4 - \frac{7}{5}c_3 + \frac{13}{16}c_2 = 0 \\ \frac{7}{16}\,(25) - \frac{7}{5}c_4 + \frac{13}{16}c_3 = 0 \end{cases} \implies \begin{bmatrix} -7/5 & 7/16 & 0 & 0 \\ 13/16 & -7/5 & 7/16 & 0 \\ 0 & 13/16 & -7/5 & 7/16 \\ 0 & 0 & 13/16 & -7/5 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} -975/16 \\ 0 \\ 0 \\ -175/16 \end{bmatrix}$$

This system of equations, $Ac = b$, can be solved directly by inverting A^{-1}. Furthermore, the structure of this system can be generalized for different Δx-values. Below is the MATLAB code to solve these systems of equations for different parameters:

```
% parameters
D = 2.5; U = 0.75; k = 0.15; dx = 2; c0 = 75; c10 = 25;

% solving Ac=b, for square matrix A of size n and
% column vector b of size n

n = 10/dx - 1;  % matrix size, n by n & # of unknowns & eqns
A = zeros(n);   % instantiate an empty matrix

% calculate coefficients for c_{i+1}, c_i, & c_{i-1}
cIPlusOne = D/dx^2  - U/(2*dx);
cI = -2*D/dx^2 - k;
cIMinusOne = D/dx^2 + U/(2*dx);

% fill out first & last row of matrix A
A(1,1) = cI;
A(1,2) = cIPlusOne;
A(n,n) = cI;
A(n,n-1) = cIMinusOne;
% fill out the rest of matrix A
for r = 2:n-1
    A(r,r) = cI;
    A(r,r-1) = cIMinusOne;
    A(r,r+1) = cIPlusOne;
end

% instantiate & fill out matrix b
b = zeros(n,1);
b(1,1) = -cIMinusOne*c0;
b(n,1) = -cIPlusOne*c10;

% display system of equations & solutions for concentrations
A, b, c = [c0; inv(A)*b; c10]

% plot
f1 = figure();
    plot(0:dx:10, c, 'LineWidth', 3, 'Color', 'r');
    grid;
    title('Concentration Estimate for \Delta x = 0.1', 'FontSize',
       12)
```

For $\Delta x = 2$, 1, and 0.1, the plots for concentrations at different distances are as follows:

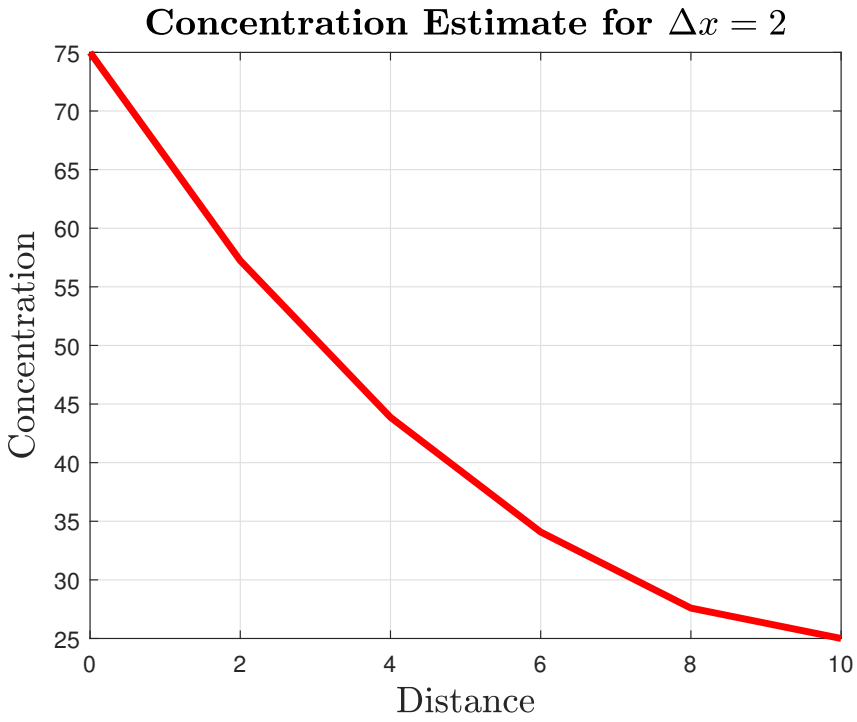

Figure 1: Plot of the estimated concentrations for $\Delta x = 2$.

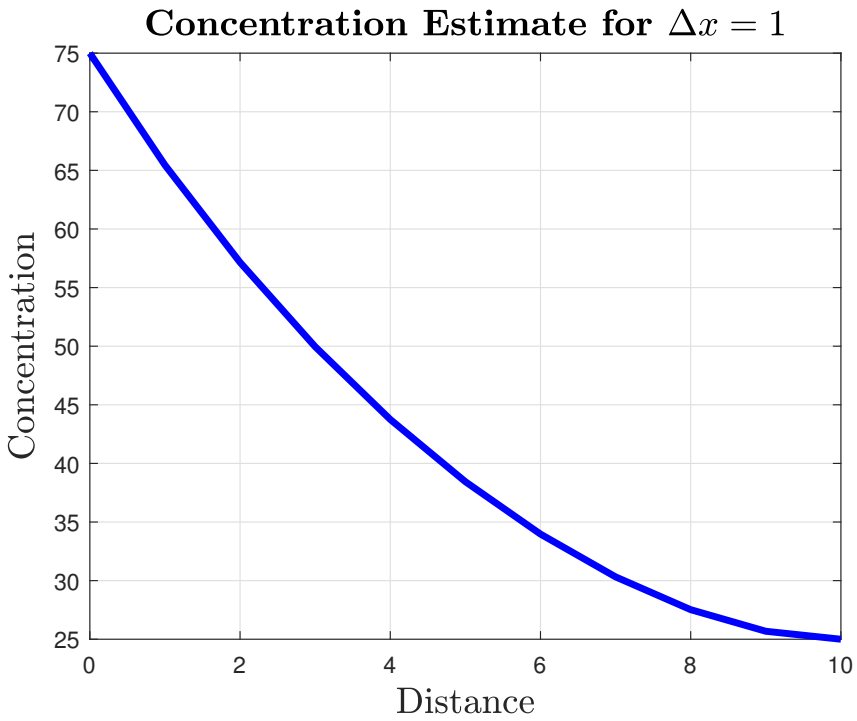

Figure 2: Plot of the estimated concentrations for $\Delta x = 1$.

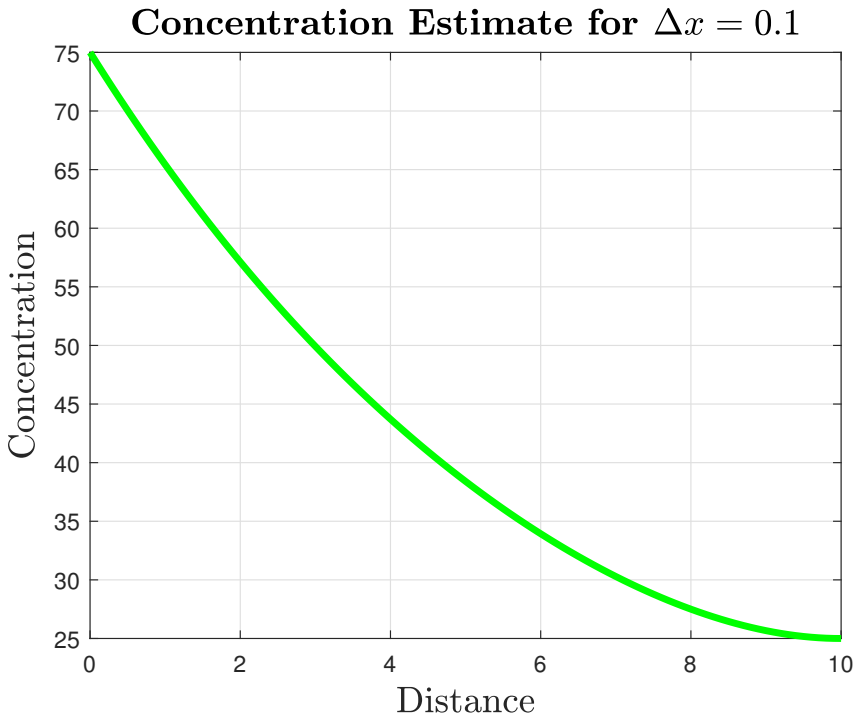

Figure 3: Plot of the estimated concentrations for $\Delta x = 0.1$.

Notice that as Δx decreases, the graph of the concentrations gets closer in shape to an exponential decay.

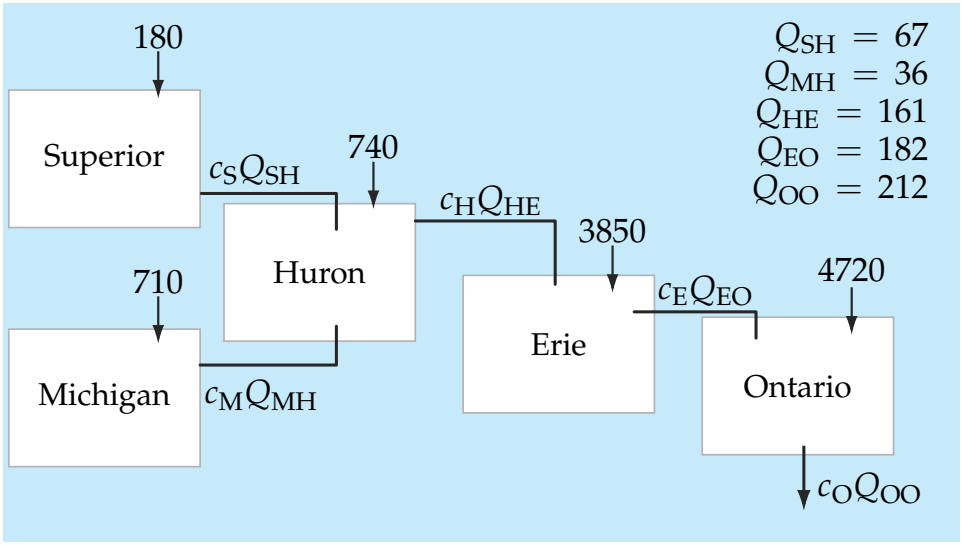

Figure 4: Abstract description of the Chloride mass balance and flows for the Great Lakes.

5. Fig. 4 shows the Chloride mass balance for the Great Lakes in terms of the flows. Determine the concentration of chloride in each of the Great Lakes $c_{1,2,3,4,5}$ for the five lakes using the information shown in Fig. 4. The flow rates are given in the legend of the figure. Compute the matrix inverse, and use it to determine the percent reduction of the concentration in Lake Ontario due to a 50% reduction of the loadings to Lake Superior and Lake Michigan.

   To solve the problem, you need to know that the flow rates Q (in cubic meters per minute) and concentrations (in milligrams per cubic meter) are related through simple mass-balance models:

$$\sum q_{\text{in}} c_{\text{in}} = \sum q_{\text{out}} c_{\text{out}}.$$

Problem 5 Solution: Set up system of equations from the given info and solve for the concentrations in mg/m^3:

$$\begin{aligned} 180 &= Q_{SH} c_S \\ 710 &= Q_{MH} c_M \\ 740 + Q_{SH} c_S + Q_{MH} c_M &= Q_{HE} c_H \\ 3850 + Q_{HE} c_H &= Q_{EO} c_E \\ 4720 + Q_{EO} c_E &= Q_{OO} c_O \end{aligned} \qquad \begin{aligned} Q_{SH} &= 67 \\ Q_{MH} &= 36 \\ Q_{HE} &= 161 \\ Q_{EO} &= 182 \\ Q_{OO} &= 212 \end{aligned}$$

$$\begin{cases} 180 = 67 c_S \\ 710 = 36 c_M \\ 740 + 67 c_S + 36 c_M = 161 c_H \\ 3850 + 161 c_H = 182 c_E \\ 4720 + 182 c_E = 212 c_O \end{cases} \implies \begin{cases} 67 c_S = 180 \\ 36 c_M = 710 \\ 67 c_S + 36 c_M - 161 c_H = -740 \\ 161 c_H - 182 c_E = -3850 \\ 182 c_E - 212 c_O = -4720 \end{cases}$$

$$\implies \begin{bmatrix} 67 & 0 & 0 & 0 & 0 \\ 0 & 36 & 0 & 0 & 0 \\ 67 & 36 & -161 & 0 & 0 \\ 0 & 0 & 161 & -182 & 0 \\ 0 & 0 & 0 & 182 & -212 \end{bmatrix} \begin{bmatrix} c_S \\ c_M \\ c_H \\ c_E \\ c_O \end{bmatrix} = \begin{bmatrix} 180 \\ 710 \\ -740 \\ -3850 \\ -4720 \end{bmatrix} \xrightarrow{R_3 \leftarrow R_3 - R_1 - R_2}$$

$$\left[\begin{array}{ccccc|c} 67 & 0 & 0 & 0 & 0 & 180 \\ 0 & 36 & 0 & 0 & 0 & 710 \\ 0 & 0 & -161 & 0 & 0 & -1630 \\ 0 & 0 & 161 & -182 & 0 & -3850 \\ 0 & 0 & 0 & 182 & -212 & -4720 \end{array}\right] \xrightarrow{R_4 \leftarrow R_4 + R_3}$$

$$\left[\begin{array}{ccccc|c} 67 & 0 & 0 & 0 & 0 & 180 \\ 0 & 36 & 0 & 0 & 0 & 710 \\ 0 & 0 & -161 & 0 & 0 & -1630 \\ 0 & 0 & 0 & -182 & 0 & -5480 \\ 0 & 0 & 0 & 182 & -212 & -4720 \end{array}\right] \xrightarrow{R_5 \leftarrow R_5 + R_4}$$

$$\left[\begin{array}{ccccc|c} 67 & 0 & 0 & 0 & 0 & 180 \\ 0 & 36 & 0 & 0 & 0 & 710 \\ 0 & 0 & -161 & 0 & 0 & -1630 \\ 0 & 0 & 0 & -182 & 0 & -5480 \\ 0 & 0 & 0 & 0 & -212 & -10200 \end{array}\right]$$

$$\implies \begin{bmatrix} c_S \\ c_M \\ c_H \\ c_E \\ c_O \end{bmatrix} = \begin{bmatrix} 180/67 \\ 710/36 \\ -1630/-161 \\ -5480/-182 \\ -10200/-212 \end{bmatrix} \approx \begin{bmatrix} 2.686567 \\ 19.7\overline{2} \\ 10.124224 \\ 30.109890 \\ 48.113208 \end{bmatrix}$$

Following the same row operations and normalizing each row results with I_5 as the augmented matrix results in the inverse of the coefficients matrix:

$$\implies A^{-1} = \begin{bmatrix} 1/67 & 0 & 0 & 0 & 0 \\ 0 & 1/36 & 0 & 0 & 0 \\ 1/161 & 1/161 & -1/161 & 0 & 0 \\ 1/182 & 1/182 & -1/182 & -1/182 & 0 \\ 1/212 & 1/212 & -1/212 & -1/212 & -1/212 \end{bmatrix}$$

With a 50% reduction of the loadings to Lake Superior and Lake Michigan, the b matrix becomes:

$$\begin{bmatrix} 180/2 \\ 710/2 \\ -740 \\ -3850 \\ -4720 \end{bmatrix}$$

and the updated concentrations in $\mathrm{mg/m^3}$ for each lake are as follows:

$$c = A^{-1}b = \begin{bmatrix} 1/67 & 0 & 0 & 0 & 0 \\ 0 & 1/36 & 0 & 0 & 0 \\ 1/161 & 1/161 & -1/161 & 0 & 0 \\ 1/182 & 1/182 & -1/182 & -1/182 & 0 \\ 1/212 & 1/212 & -1/212 & -1/212 & -1/212 \end{bmatrix} \begin{bmatrix} 180/2 \\ 710/2 \\ -740 \\ -3850 \\ -4720 \end{bmatrix}$$

$$\implies c = \begin{bmatrix} c_S \\ c_M \\ c_H \\ c_E \\ c_O \end{bmatrix} = \begin{bmatrix} 90/67 \\ 355/36 \\ 1185/161 \\ 5035/182 \\ 9755/212 \end{bmatrix} \approx \begin{bmatrix} 1.343284 \\ 9.86\overline{1} \\ 7.360248 \\ 27.664835 \\ 46.014151 \end{bmatrix}$$

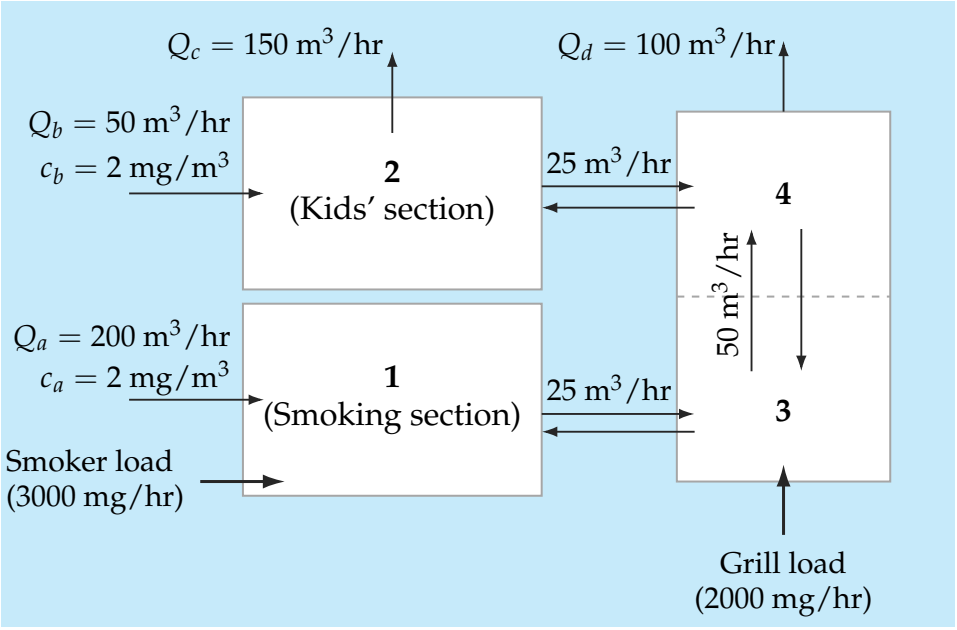

Figure 5: Overhead view of rooms in a restaurant. The one-way arrows represent volumetric airflows, whereas the two-way arrows represent diffusive mixing. The smoker and grill loads add carbon monoxide mass to the system but negligible airflow.

6. Indoor air pollution deals with air contamination in enclosed spaces such as homes, offices, work areas, etc. Suppose that you are designing a ventilation system for a restaurant as shown in Fig. 5. The restaurant serving area consists of two square rooms and one elongated room. Room 1 and room 3 have sources of carbon monoxide from smokers and a faulty grill, respectively. Steady-state mass balances can be written for each room. For example, for the smoking section (room 1), the balance can be written as

$$0 = \underbrace{W_{\text{smoker}}}_{\text{load}} + \underbrace{Q_a c_a}_{\text{inflow}} - \underbrace{Q_a c_1}_{\text{outflow}} + \underbrace{E_{13}(c_3 - c_1)}_{\text{mixing}}$$

or substituting the parameters

$$225c_1 - 25c_3 = 2400.$$

Similar balances can be written for the other rooms.

(a) Solve for the steady-state concentration of carbon monoxide in each room.

(b) Determine what percent of the carbon monoxide in the kids' section is due to (i) the smokers, (ii) the grill, and (iii) the air in the intake vents.

(c) If the smoker and grill loads are increased to 2000 and 5000 mg/hr, respectively, use the matrix inverse to determine the increase in the concentration in the kids' section.

(d) How does the concentration in the kids' area change if a screen is constructed so that the mixing between areas 2 and 4 is decreased to 5 m^3/hr?

Problem 6 Solution:

(a) Construct system of equations:

$$3000 + (200)(2) - 200c_1 + 25(c_3 - c_1) = 0 \quad \implies 225c_1 - 25c_3 = 3400$$

$$(50)(2) + 25c_4 - 150c_2 + 25(c_4 - c_2) = 0 \quad \implies -175c_2 + 50c_4 = -100$$

$$2000 + 25c_1 + 25(c_1 - c_3) + 50(c_4 - c_3) - 50c_3 = 0 \quad \implies 50c_1 - 125c_3 + 50c_4 = -2000$$

$$50c_3 + 50(c_3 - c_4) + 25(c_2 - c_4) - 25c_4 - 100c_4 = 0 \quad \implies 25c_2 + 100c_3 - 200c_4 = 0$$

Solve the system for the concentrations c (in mg/m^3) of carbon monoxide in each room:

$$Ac = b \implies \begin{bmatrix} 225 & 0 & -25 & 0 \\ 0 & -175 & 0 & 50 \\ 50 & 0 & -125 & 50 \\ 0 & 25 & 100 & -200 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 3400 \\ -100 \\ -2000 \\ 0 \end{bmatrix}$$

$$c = A^{-1}b \implies \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \approx \begin{bmatrix} 18.3894 \\ 4.9637 \\ 29.5050 \\ 15.3729 \end{bmatrix}$$

(b) i. Smokers:
Set $b = b_{\text{smoker}} = \begin{bmatrix} 3000 & 0 & 0 & 0 \end{bmatrix}^\top$ and solve $Ac_{\text{smoker}} = b_{\text{smoker}}$:

$$c_{\text{smoker}} = \begin{bmatrix} c_{1,\text{ smoker}} \\ c_{2,\text{ smoker}} \\ c_{3,\text{ smoker}} \\ c_{4,\text{ smoker}} \end{bmatrix} \approx \begin{bmatrix} 14.1254 \\ 1.0561 \\ 7.1287 \\ 3.6964 \end{bmatrix}$$

Compute percent in room 2:

$$c_{2,\text{ smoker }\%} = \frac{1.0561}{c_2} \cdot 100\% = \frac{1.0561}{4.9637} \cdot 100\% \quad \implies c_{2,\text{ smoker }\%} \approx 21.2766\%$$

ii. Grill:
Set $b = b_{\text{grill}} = \begin{bmatrix} 0 & 0 & -2000 & 0 \end{bmatrix}^\top$ and solve $Ac_{\text{grill}} = b_{\text{grill}}$:

$$c_{\text{grill}} = \begin{bmatrix} c_{1,\text{ grill}} \\ c_{2,\text{ grill}} \\ c_{3,\text{ grill}} \\ c_{4,\text{ grill}} \end{bmatrix} \approx \begin{bmatrix} 2.3762 \\ 3.1683 \\ 21.3861 \\ 11.0891 \end{bmatrix}$$

Compute percent in room 2:

$$c_{2,\text{ grill }\%} = \frac{3.1683}{c_2} \cdot 100\% = \frac{3.1683}{4.9637} \cdot 100\% \quad \implies c_{2,\text{ grill }\%} \approx 63.8298\%$$

iii. Air vents:
Set $b = b_{\text{vents}} = \begin{bmatrix} 400 & -100 & 0 & 0 \end{bmatrix}^\top$ and solve $Ac_{\text{vents}} = b_{\text{vents}}$:

$$c_{\text{vents}} = \begin{bmatrix} c_{1,\text{ vents}} \\ c_{2,\text{ vents}} \\ c_{3,\text{ vents}} \\ c_{4,\text{ vents}} \end{bmatrix} \approx \begin{bmatrix} 1.8878 \\ 0.7393 \\ 0.9901 \\ 0.5875 \end{bmatrix}$$

Compute percent in room 2:

$$c_{2,\text{ vents }\%} = \frac{0.7393}{c_2} \cdot 100\% = \frac{0.7393}{4.9637} \cdot 100\% \quad \implies c_{2,\text{ vents }\%} \approx 14.8936\%$$

(c) If the smoker and grill loads are modified to become 2000 and 5000 mg/hr, respectively, then $b = b_{\text{modified}} = \begin{bmatrix} 2400 & -100 & -3000 & 0 \end{bmatrix}^\top$. Therefore,

$$c_{\text{modified}} = A^{-1} b_{\text{modified}} \approx \begin{bmatrix} 17.2453 \\ 9.3641 \\ 59.2079 \\ 30.7745 \end{bmatrix}$$

and the increase in concentration in room 2 is $c_{\text{modified}} - c_2 \approx 9.3641 - 4.9637 \approx 4.4004$ mg/m^3.

(d) If a screen is constructed so that the mixing between areas 2 and 4 is decreased to 5 m^3/hr, the 2nd and 4th equations in the original system are changed.

Equation 2 becomes:

$$(50)(2) + 5c_4 - 150c_2 + 5(c_4 - c_2) = 0 \quad \implies -145c_2 + 10c_4 = -100$$

and equation 4 becomes:

$$50c_3 + 50(c_3 - c_4) + 5(c_2 - c_4) - 5c_4 - 100c_4 = 0 \quad \implies 5c_2 + 100c_3 - 160c_4 = 0.$$

Solving the new system of equations yields:

$$A_{\text{new}} c_{\text{new}} = b \implies \begin{bmatrix} 225 & 0 & -25 & 0 \\ 0 & -145 & 0 & 10 \\ 50 & 0 & -125 & 50 \\ 0 & 5 & 100 & -160 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 3400 \\ -100 \\ -2000 \\ 0 \end{bmatrix}$$

$$c_{\text{new}} = A_{\text{new}}^{-1} b \implies \begin{bmatrix} c_{1,\text{ new}} \\ c_{2,\text{ new}} \\ c_{3,\text{ new}} \\ c_{4,\text{ new}} \end{bmatrix} \approx \begin{bmatrix} 18.5867 \\ 2.0423 \\ 31.2803 \\ 19.6140 \end{bmatrix}$$

Then, the concentration in room 2 increases by $c_{2,\text{ new}} - c_2 \approx 2.0423 - 4.9637 \approx -2.9214$ mg/m^3 (meaning the concentration decreases by 2.9214 mg/m^3). My MATLAB script code for problem 8 is produced below:

```
% part A
A = [225,0,-25,0;0,-175,0,50;50,0,-125,50;0,25,100,-200]
b = [3400;-100;-2000;0]

c = inv(A)*b

% part B
b_smoker = [3000;0;0;0];
c_smoker = inv(A)*b_smoker;
c_2SmokerPercent = c_smoker(2)/c(2) .* 100

b_grill = [0;0;-2000;0];
c_grill = inv(A)*b_grill;
c_2GrillPercent = c_grill(2)/c(2) .* 100

b_vents = [400;-100;0;0];
c_vents = inv(A)*b_vents;
c_2ventsPercent = c_vents(2)/c(2) .* 100

% part C
b_modified = b + [-1000;0;-3000;0];
c_modified = inv(A)*b_modified;
c_2Increased = c_modified(2) - c(2)

% part D
A_new = [225,0,-25,0;0,-145,0,10;50,0,-125,50;0,5,100,-160];
c_new = inv(A_new)*b
c_2New = c_new(2);
change_in_c_2 = c_new(2) - c(2)
```

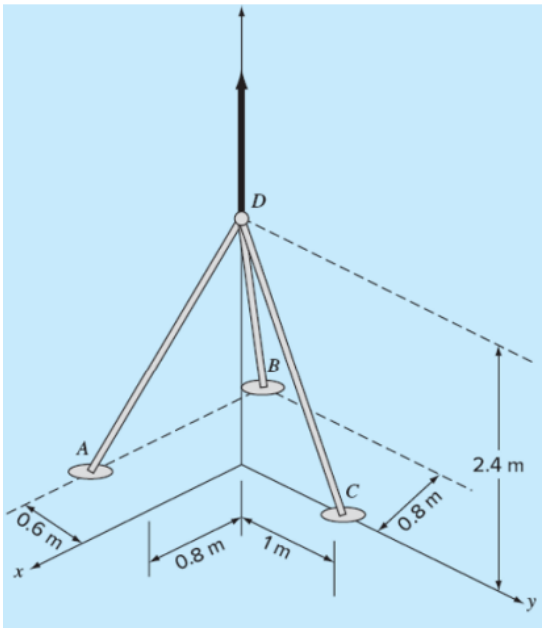

Figure 6: The uncomfortable, virtual forces acting on a tripod while an influencer takes pictures and performs obnoxious dance moves.

7. An upward force of 20 kN is applied at the top of a tripod as depicted in Fig. 6. Determine the forces in the legs of the tripod via formulating a linear system of equations $Ax = b$ and solving it via Gaussian elimination **manually** and the SOR method with optimal ω for only 5 iterations. Compare the results.

   To solve this problem, you can assume that each leg is in tension, which mean that each pulls on point D.

   Problem 7 Solution: From the tripod figure:

$$\mathbf{A} = \langle 0.8, -0.6, 0\rangle, \quad \mathbf{B} = \langle -0.8, -0.6, 0\rangle, \quad \mathbf{C} = \langle 0, 1, 0\rangle, \quad \mathbf{D} = \langle 0, 0, 2.4\rangle$$

   Therefore,

$$\overrightarrow{DA} = \langle 0.8, -0.6, -2.4\rangle \implies \left|\left|\overrightarrow{DA}\right|\right| = \sqrt{(0.8)^2 + (-0.6)^2 + (-2.4)^2} = 2.6$$
$$\overrightarrow{DB} = \langle -0.8, -0.6, -2.4\rangle \implies \left|\left|\overrightarrow{DB}\right|\right| = \sqrt{(-0.8)^2 + (-0.6)^2 + (-2.4)^2} = 2.6$$
$$\overrightarrow{DC} = \langle 0, 1, -2.4\rangle \implies \left|\left|\overrightarrow{DC}\right|\right| = \sqrt{(0)^2 + (1)^2 + (-2.4)^2} = 2.6$$

$$\overrightarrow{F_{DA}} = F_{DA}\langle 4/13, -3/13, -12/13\rangle$$
$$\overrightarrow{F_{DB}} = F_{DB}\langle -4/13, -3/13, -12/13\rangle$$
$$\overrightarrow{F_{DC}} = F_{DC}\langle 0, 5/13, -12/13\rangle$$

   Sum forces in $x, y,$ and z:

$$\sum F_x = 0 \implies \frac{4}{13}F_{DA} - \frac{4}{13}F_{DB} + 0F_{DC} = 0$$
$$\sum F_y = 0 \implies -\frac{3}{13}F_{DA} - \frac{3}{13}F_{DB} + \frac{5}{13}F_{DC} = 0$$
$$\sum F_z = 0 \implies -\frac{12}{13}F_{DA} - \frac{12}{13}F_{DB} - \frac{12}{13}F_{DC} + 20 = 0$$

(a) Solve system of equations using Gaussian elimination:

$$\begin{bmatrix} 4/13 & -4/13 & 0 \\ -3/13 & -3/13 & 5/13 \\ -12/13 & -12/13 & -12/13 \end{bmatrix} \begin{bmatrix} F_{DA} \\ F_{DB} \\ F_{DC} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -20 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 + \frac{3}{4}R_1}$$

$$\left[\begin{array}{ccc|c} 4/13 & -4/13 & 0 & 0 \\ 0 & -6/13 & 5/13 & 0 \\ -12/13 & -12/13 & -12/13 & -20 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 + 3R_1}$$

$$\left[\begin{array}{ccc|c} 4/13 & -4/13 & 0 & 0 \\ 0 & -6/13 & 5/13 & 0 \\ 0 & -24/13 & -12/13 & -20 \end{array}\right] \xrightarrow{R_3 \leftarrow R_3 - 4R_2}$$

$$\left[\begin{array}{ccc|c} 4/13 & -4/13 & 0 & 0 \\ 0 & -6/13 & 5/13 & 0 \\ 0 & 0 & -32/13 & -20 \end{array}\right]$$

$$\begin{cases} -\frac{32}{13}F_{DC} = -20 & \implies F_{DC} = 8.125\,[\text{kN}] \\ -\frac{6}{13}F_{DB} + \frac{5}{13}F_{DC} = 0 & \implies F_{DB} = 325/48 \approx 6.771\,[\text{kN}] \\ \frac{4}{13}F_{DA} - \frac{4}{13}F_{DB} = 0 & \implies F_{DA} = 325/48 \approx 6.771\,[\text{kN}] \end{cases}$$

(b) Solve system of equations using SOR w/ optimal ω:

$$D = \begin{bmatrix} 4/13 & 0 & 0 \\ 0 & -3/13 & 0 \\ 0 & 0 & -12/13 \end{bmatrix} \quad E = \begin{bmatrix} 0 & 0 & 0 \\ 3/13 & 0 & 0 \\ 12/13 & 12/13 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 4/13 & 0 \\ 0 & 0 & -5/13 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\omega^* = \frac{2}{1 + \sqrt{1 - \left(\rho\left(D^{-1}\left(E+F\right)\right)\right)^2}}$$

$$D^{-1} = \begin{bmatrix} 13/4 & 0 & 0 \\ 0 & -13/3 & 0 \\ 0 & 0 & -13/12 \end{bmatrix} \qquad E + F = \begin{bmatrix} 0 & 4/13 & 0 \\ 3/13 & 0 & -5/13 \\ 12/13 & 12/13 & 0 \end{bmatrix}$$

Compute $\rho\left(D^{-1}\left(E+F\right)\right)$:

$$D^{-1}\left(E+F\right) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 5/3 \\ -1 & -1 & 0 \end{bmatrix}$$

$$\det\left(D^{-1}\left(E+F\right) - \lambda I_3\right) = 0$$

$$\begin{vmatrix} -\lambda & 1 & 0 \\ -1 & -\lambda & 5/3 \\ -1 & -1 & -\lambda \end{vmatrix} = 0$$

$$-\lambda\left(\lambda^2 + \frac{5}{3}\right) - 1\left(\lambda + \frac{5}{3}\right) + 0 = 0$$

$$\implies \lambda \approx -0.5594, 0.2797 \pm 1.7033i$$

$$\implies \rho\left(D^{-1}\left(E+F\right)\right) \approx \sqrt{(0.2797)^2 + (1.7033)^2} \approx 1.726$$

$$\implies \text{imaginary } \omega^* \implies \text{cannot use SOR with optimal } \omega$$

8. For the following system of equations $Ax = b$

$$3x_1 - x_2 + x_3 = 1, \;\; 3x_1 + 6x_2 + 2x_3 = 0, \;\; 3x_1 + 3x_2 + 7x_3 = 4$$

use the Richardson, Jacobi, Gauss-Seidel, and SOR (with $\omega = 0.5, 1.1, 1.8$) to compute the solution $x^* = A^{-1}b$ for ten iterations. To do so, you need to develop codes for these four methods—you will be using this code in some other problems.

You will also need to show the matrices M and N (D, E, F) for these methods.

Plot the performance of the four methods as a function of the number of iterations and the error norm $||x^* - x^{(i)}||$ where i the iteration number.

Problem 8 Solution: Given:

$$A = \begin{bmatrix} 3 & -1 & 1 \\ 3 & 6 & 2 \\ 3 & 3 & 7 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 0 \\ 4 \end{bmatrix}$$
$$\implies Ax = b$$

MATLAB code to solve the above system of equations using all the methods for 10 iterations is listed below:

```
function [xStar, richardson, jacobi, gaussSeidel, SOR05, SOR11,
    SOR18] = compareAllMethods(A, b, maxIter)

xStar = inv(A)*b;

% Richardson method
In = zeros(size(A));
for i = 1:length(A)
    In(i,i) = 1;
end

% display M and N for Richardson
M_richardson = In
N_richardson = In - A

rhoT = abs(max(eig(In-A)));
if rhoT < 1
    alpha = 1;
else
    alpha = 1/rhoT;
end

xOldR = zeros(length(A),1);
xNewR = zeros(length(A),1);
errorR = zeros(1, maxIter);

for i = 1:maxIter
    xNewR = xOldR + alpha.*(b - A*xOldR);
    errorR(1,i) = norm(xStar - xNewR);
    xOldR = xNewR;
end

richardson = xNewR;


% Relaxation methods
```

```
% set D, E, and F
D = zeros(size(A));
for i = 1:length(A)
    D(i,i) = A(i,i);
end

E = zeros(size(A));
for row = 2:length(A)
    for col = 1:row-1
        E(row,col) = -A(row,col);
    end
end

F = zeros(size(A));
for row = 1:length(A)
    for col = row+1:length(A)
        F(row,col) = -A(row,col);
    end
end

% display D, E, and F
D,E,F

% Jacobi method
% display M and N for Jacobi
M_jacobi = D
N_jacobi = E+F

xOldJ = zeros(length(A),1);
xNewJ = zeros(length(A),1);
errorJ = zeros(1,maxIter);
T_j = inv(D)*(E+F);
secondTerm_j = inv(D)*b;

for i = 1:maxIter
    xNewJ = T_j*xOldJ + secondTerm_j;
    errorJ(1,i) = norm(xStar - xNewJ);
    xOldJ = xNewJ;
end

jacobi = xNewJ;

% Gauss-Seidel method
% display M and N for GS
M_gs = D-E
N_gs = F

xOldGS = zeros(length(A),1);
xNewGS = zeros(length(A),1);
errorGS = zeros(1,maxIter);
T_gs = inv(D-E)*F;
secondTerm_gs = inv(D-E)*b;

for i = 1:maxIter
    xNewGS = T_gs*xOldGS + secondTerm_gs;
    errorGS(1,i) = norm(xStar - xNewGS);
```

```
    xOldGS = xNewGS;
end

gaussSeidel = xNewGS;

% SOR method with w = 0.5
% display M and N for SOR05
M_sor05 = (1/0.5)*D-E
N_sor05 = ((1-0.5)/0.5)*D+F

xOldSOR05 = zeros(length(A),1);
xNewSOR05 = zeros(length(A),1);
errorSOR05 = zeros(1,maxIter);
T_sor05 = inv(D-0.5.*E)*((1-0.5).*D+0.5.*F);
secondTerm_sor05 = inv(D-0.5.*E)*(0.5.*b);

for i = 1:maxIter
    xNewSOR05 = T_sor05*xOldSOR05 + secondTerm_sor05;
    errorSOR05(1,i) = norm(xStar - xNewSOR05);
    xOldSOR05 = xNewSOR05;
end

SOR05 = xNewSOR05;

% SOR method with w = 1.1
% display M and N for SOR11
M_sor11 = (1/1.1)*D-E
N_sor11 = ((1-1.1)/1.1)*D+F

xOldSOR11 = zeros(length(A),1);
xNewSOR11 = zeros(length(A),1);
errorSOR11 = zeros(1,maxIter);
T_sor11 = inv(D-1.1.*E)*((1-1.1).*D+1.1.*F);
secondTerm_sor11 = inv(D-1.1.*E)*(1.1.*b);

for i = 1:maxIter
    xNewSOR11 = T_sor11*xOldSOR11 + secondTerm_sor11;
    errorSOR11(1,i) = norm(xStar - xNewSOR11);
    xOldSOR11 = xNewSOR11;
end

SOR11 = xNewSOR11;

% SOR method with w = 1.8
% display M and N for SOR18
M_sor18 = (1/1.8)*D-E
N_sor18 = ((1-1.8)/1.8)*D+F

xOldSOR18 = zeros(length(A),1);
xNewSOR18 = zeros(length(A),1);
errorSOR18 = zeros(1,maxIter);
T_sor18 = inv(D-1.8.*E)*((1-1.8).*D+1.8.*F);
secondTerm_sor18 = inv(D-1.8.*E)*(1.8.*b);

for i = 1:maxIter
    xNewSOR18 = T_sor18*xOldSOR18 + secondTerm_sor18;
```

```
    errorSOR18(1,i) = norm(xStar - xNewSOR18);
    xOldSOR18 = xNewSOR18;
end

SOR18 = xNewSOR18;


% plot the errors for each method
xValues = 1:maxIter;

f1 = figure();
    plot(xValues, errorR, 'LineWidth', 3, 'Color', 'r');
    hold on;
    plot(xValues, errorJ, 'LineWidth', 3, 'Color', 'b');
    plot(xValues, errorGS, 'LineWidth', 3, 'Color', 'g');
    plot(xValues, errorSOR05, 'LineWidth', 3, 'Color', 'm');
    plot(xValues, errorSOR11, 'LineWidth', 3, 'Color', 'k');
    set(gca, 'FontSize', 12);
    grid;
    title(''\textbf{Convergence Plot}', 'FontSize', 16, '
       Interpreter', 'latex');
    xlabel('Number of Iterations', 'FontSize', 16, 'Interpreter', '
       latex');
    ylabel('Error Norm $\left|\left|x^*-x^{\left(i\right)}\right|\
       right|$', 'FontSize', 16, 'Interpreter', 'latex');
    legend('Richardson', 'Jacobi', 'Gauss-Seidel', ...
        'SOR $\left(\omega=0.5\right)$', ...
        'SOR $\left(\omega=1.1\right)$', ...
        'Location', 'best', ...
        'Interpreter', 'latex')

f2 = figure();
   plot(xValues, errorSOR18, 'LineWidth', 3, 'Color', 'c');
    set(gca, 'FontSize', 12);
    grid;
    title('\textbf{Divergence Plot of SOR}', '\textbf{Method with
       $\omega=1.8$}', 'FontSize', 16, 'Interpreter', 'latex');
    xlabel('Number of Iterations', 'FontSize', 16, 'Interpreter', '
       latex');
    ylabel('Error Norm $\left|\left|x^*-x^{\left(i\right)}\right|\
       right|$', 'FontSize', 16, 'Interpreter', 'latex');
    legend('SOR $\left(\omega=1.8\right)$', ...
        'Location', 'best', ...
        'Interpreter', 'latex')
```

The output for the above code is below, along with the convergence (Figure 7) and divergence (Figure 8) plots.

```
[xStar, richardson, jacobi, gaussSeidel, SOR05, SOR11, SOR18] =
    compareAllMethods(A, b, 10)

M_richardson =

     1     0     0
     0     1     0
     0     0     1


N_richardson =

    -2     1    -1
    -3    -5    -2
    -3    -3    -6


D =

     3     0     0
     0     6     0
     0     0     7


E =

     0     0     0
    -3     0     0
    -3    -3     0


F =

     0     1    -1
     0     0    -2
     0     0     0


M_jacobi =

     3     0     0
     0     6     0
     0     0     7


N_jacobi =

     0     1    -1
    -3     0    -2
    -3    -3     0


M_gs =

```

```
     3     0     0
     3     6     0
     3     3     7


N_gs =

     0     1    -1
     0     0    -2
     0     0     0


M_sor05 =

     6     0     0
     3    12     0
     3     3    14


N_sor05 =

     3     1    -1
     0     6    -2
     0     0     7


M_sor11 =

    2.7273         0         0
    3.0000    5.4545         0
    3.0000    3.0000    6.3636


N_sor11 =

   -0.2727    1.0000   -1.0000
         0   -0.5455   -2.0000
         0         0   -0.6364


M_sor18 =

    1.6667         0         0
    3.0000    3.3333         0
    3.0000    3.0000    3.8889


N_sor18 =

   -1.3333    1.0000   -1.0000
         0   -2.6667   -2.0000
         0         0   -3.1111


xStar =

```

```
    0.0351
   -0.2368
    0.6579


richardson =

    0.0397
   -0.2408
    0.6569


jacobi =

    0.0351
   -0.2369
    0.6578


gaussSeidel =

    0.0351
   -0.2368
    0.6579


SOR05 =

    0.0464
   -0.2369
    0.6476


SOR11 =

    0.0351
   -0.2368
    0.6579


SOR18 =

  -96.6028
  128.0567
  -42.1504
```

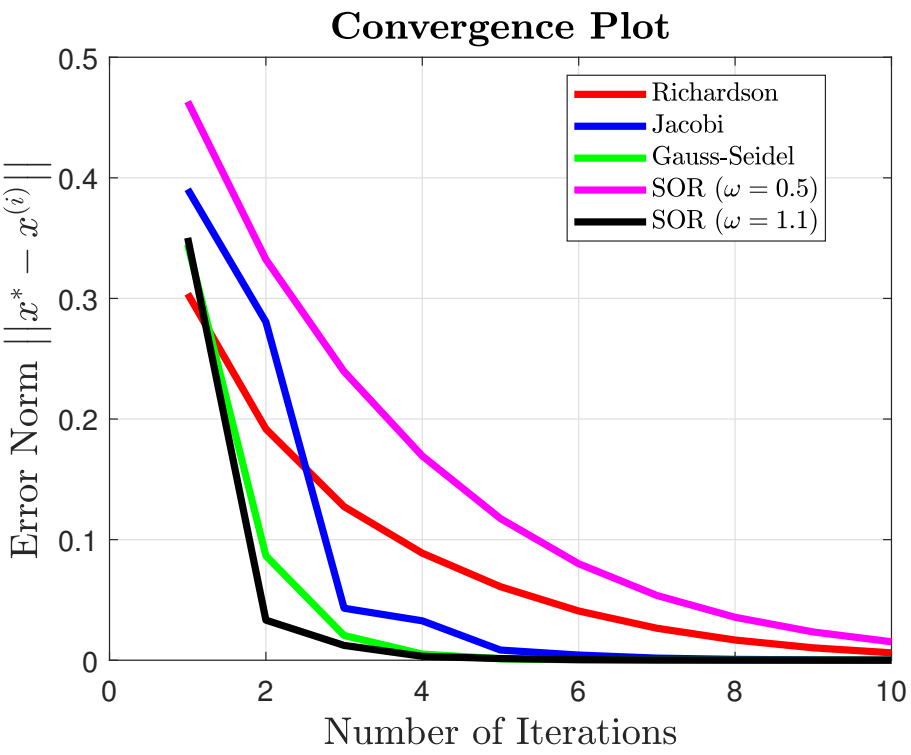

Figure 7: Convergence plot of the methods that converge.

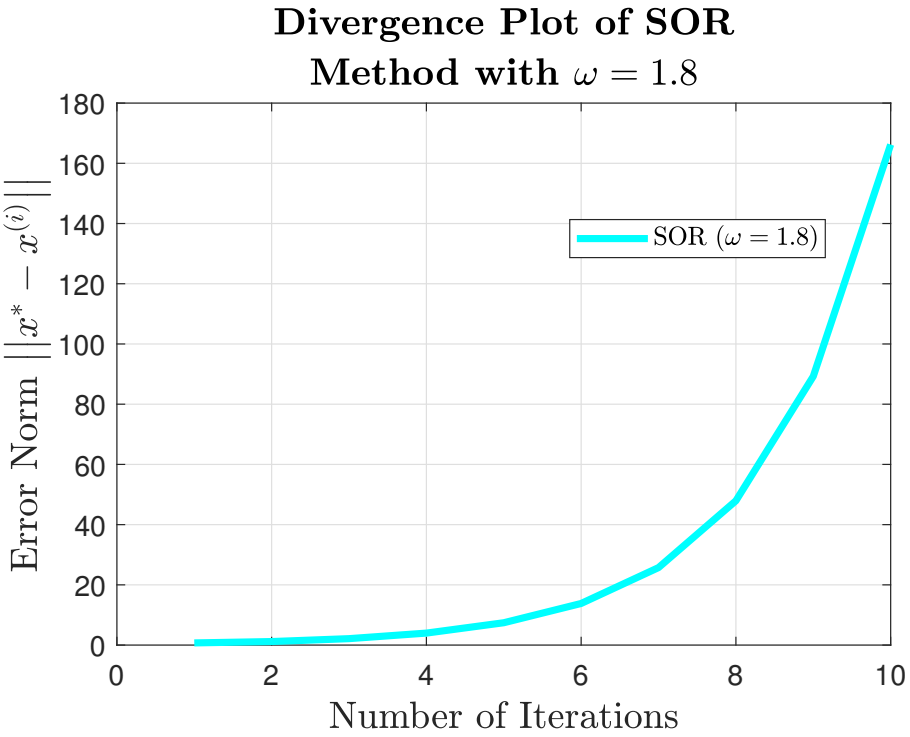

Figure 8: Divergence plot of the SOR method with $\omega = 1.8$.

9. In this problem, we wish to test the scalability of the iterative methods to solve $Ax = b$.

   (a) Generate a random square matrix A in dimension $n = 10,000$ and a random vector $b \in \mathbb{R}^n$. You can do that via matlab's `randn` command. You have to ensure that the generated matrix is indeed invertible (we talked about how to do so in class) so that the random system of equations $Ax = b$ has a unique solution.

   Suggestion: If I were you, I would start with an $n = 5$ or $n = 10$ random system. This makes it easier to debug the code and see what's happening. Then when you're sure that your code is correct, you can simply change n from 5 to 10,000.

   (b) Compute the mapping matrix T for each of the three methods (Jacobi, Gauss-Seidel, and SOR) via extracting the entries from A. Show your code.

   (c) Compute the spectral radii for the transformation matrix T for the three methods. If any of the methods yield a $\rho(T) > 1$, find a way to generate a random matrix with a spectral radius smaller than one.

   (d) Compute the optimal ω^* for the SOR method.

   (e) Compute the solution of $Ax = b$ via $x^* = A^{-1}b$ or `x = A / b` or `x = mldivide(A,b)` on Matlab. Are there any differences?

   (f) Perform 100 iterations of the three methods [Jacobi, Gauss-Seidel, and SOR (with $\omega = 1.1$ and $\omega = \omega^*$)] and discuss the results. Include your code and showcase a plot of convergence. Compare the solution with $x^* = A^{-1}b$.

   (g) Is there a computational time difference between the three methods? Which one takes the longest to execute 100 iterations?

Problem 9 Solution: The MATLAB code is produced below:

```
% problem 9

clc; clear;

% set omega = 1.1 (will use later) and maxIter = 100
omega = 1.1;
iter = 100;

% part a
% generate a random square n-by-n matrix A that is strictly diagonally
    dominant
    % this ensures that the iteration matrices converge
n = 10000;
A = rand(n);
A = A + n.*eye(n);

% generate a random n-by-one column vector b
b = rand(n,1);

% part b
% set D, E, and F
D = zeros(n,n);
for i = 1:n
    D(i,i) = A(i,i);
end

E = zeros(n,n);
for row = 2:n
    for col = 1:row-1
        E(row,col) = -A(row,col);
```

```
    end
end

F = zeros(n,n);
for row = 1:n
    for col = row+1:n
        F(row,col) = -A(row,col);
    end
end

% compute mapping matrix T for Jacobi, Gauss-Seidel, & SOR
T_j = inv(D)*(E+F);
T_gs = inv(D-E)*F;
T_sor11 = inv(D-omega.*E)*((1-omega).*D+omega.*F);

% part c
% compute spectral radius, rho, for each method
rho_j = abs(eigs(T_j,1))
rho_gs = abs(eigs(T_gs,1))
rho_sor11 = abs(eigs(T_sor11,1))

% part d
% compute optimal omega and corresponding mapping matrix for SOR_opt
omega_opt = 2/(1+sqrt(1-(rho_j)^2))
T_sor_opt = inv(D-omega_opt.*E)*((1-omega_opt).*D+omega_opt.*F);

% part e
% compute true solution, x*, via A^(-1)*b, A\b, and mldivide(A,b)
xStar = inv(A)*b;
xStarTwo = A\b;
xStarThree = mldivide(A,b);

% part f and g
% 100 iterations of each method
% showcase convergence plot
% find any computational time differences between methods

% Jacobi
j_time_start = cputime;
xOldJ = zeros(n,1);
xNewJ = zeros(n,1);
errorJ = zeros(1,iter);

secondTerm_j = inv(D)*b;

for i = 1:iter
    xNewJ = T_j*xOldJ + secondTerm_j;
    errorJ(1,i) = norm(xStar - xNewJ);
    xOldJ = xNewJ;
end

jacobi = xNewJ;
j_comp_time = cputime - j_time_start

% Gauss-Seidel
gs_time_start = cputime;
```

```
xOldGS = zeros(n,1);
xNewGS = zeros(n,1);
errorGS = zeros(1,iter);

secondTerm_gs = inv(D-E)*b;

for i = 1:iter
    xNewGS = T_gs*xOldGS + secondTerm_gs;
    errorGS(1,i) = norm(xStar - xNewGS);
    xOldGS = xNewGS;
end

gaussSeidel = xNewGS;
gs_comp_time = cputime - gs_time_start

% SOR method with omega = 1.1
sor11_time_start = cputime;
xOldSOR11 = zeros(n,1);
xNewSOR11 = zeros(n,1);
errorSOR11 = zeros(1,iter);

secondTerm_SOR11 = inv(D-omega.*E)*(omega.*b);

for i = 1:iter
    xNewSOR11 = T_sor11*xOldSOR11 + secondTerm_SOR11;
    errorSOR11(1,i) = norm(xStar - xNewSOR11);
    xOldSOR11 = xNewSOR11;
end

SOR11 = xNewSOR11;
sor11_comp_time = cputime - sor11_time_start

% SOR method with omega = omega_opt
sor_opt_time_start = cputime;
xOldSOR_opt = zeros(n,1);
xNewSOR_opt = zeros(n,1);
errorSOR_opt = zeros(1,iter);

secondTerm_sor_opt = inv(D-omega_opt.*E)*(omega_opt.*b);

for i = 1:iter
    xNewSOR_opt = T_sor_opt*xOldSOR_opt + secondTerm_sor_opt;
    errorSOR_opt(1,i) = norm(xStar - xNewSOR_opt);
    xOldSOR_opt = xNewSOR_opt;
end

SOR_opt = xNewSOR_opt;
sor_opt_comp_time = cputime - sor_opt_time_start

% plot convergence of each method separately and all together
xValues = 1:iter;
f1 = figure();
    plot(xValues, errorJ, 'LineWidth', 3, 'Color', 'b');
    hold on;
    plot(xValues, errorGS, 'LineWidth', 3, 'Color', 'g');
    plot(xValues, errorSOR11, 'LineWidth', 3, 'Color', 'm');
```

```
142     plot(xValues, errorSOR_opt, 'LineWidth', 3, 'Color', 'k');
143     set(gca, 'FontSize', 12);
144     grid;
145     title('\textbf{Convergence Plot}', '\textbf{for each Method}', '
            FontSize', 16, 'Interpreter', 'latex');
146     xlabel('Number of Iterations', 'FontSize', 16, 'Interpreter', '
            latex');
147     ylabel('Error Norm $\left|\left|x^*-x^{\left(i\right)}\right|\right
            |$', 'FontSize', 16, 'Interpreter', 'latex');
148     legend('Jacobi', 'Gauss-Seidel', ...
149         'SOR $\left(\omega=1.1\right)$', ...
150         'SOR $\left(\omega=\omega^*\right)$', ...
151         'Location', 'best', ...
152         'Interpreter', 'latex')
```

Running the above script produces the following output and plot (Figure 9):

```
rho_j =

    0.4999


rho_gs =

    0.1072


rho_sor11 =

    0.1992


omega_opt =

    1.0718


j_comp_time =

   13.5000


gs_comp_time =

   33.1406


sor11_comp_time =

   35.0156


sor_opt_comp_time =

   31.7344
```

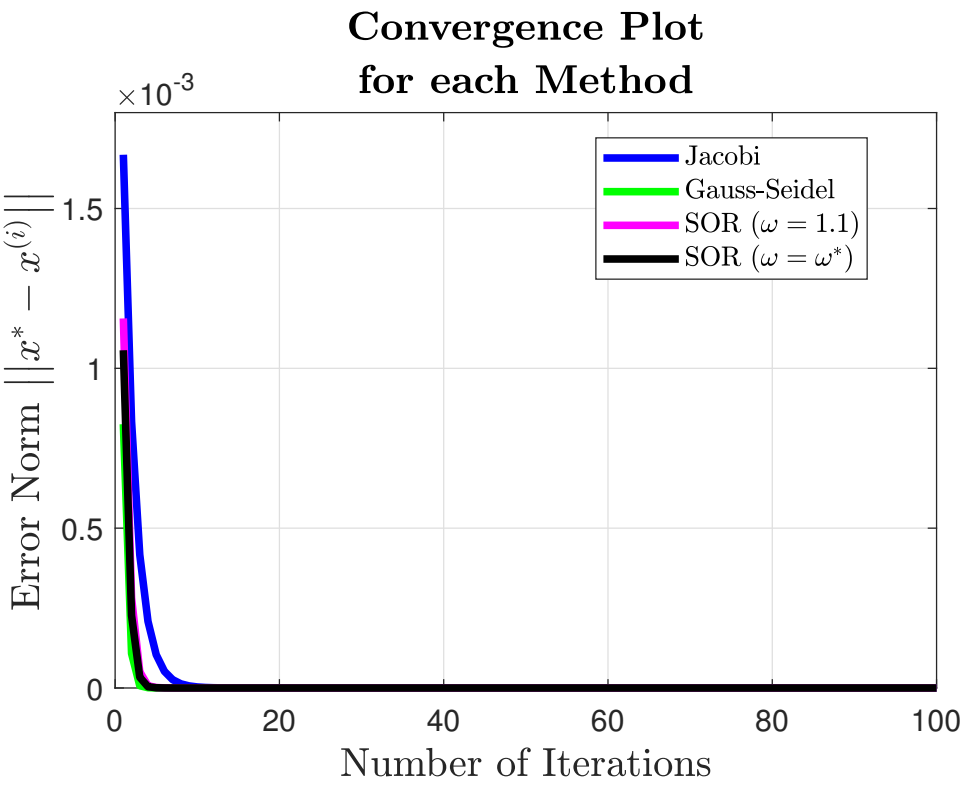

Figure 9: Convergence of each method when solving $Ax = b$ for a randomly generated 10,000-by-10,000 matrix A and a randomly generated 1-by-10,000 column vector b.

From the outputs, the SOR method with $\omega = 1.1$ takes the longest to execute 100 iterations. However, it's only slower by a couple seconds (excluding the fastest method, Jacobi).