Chapter 3 Iterative Methods for Solving Linear Systems

3.1 General Considerations

Direct and iterative methods offer two distinct approaches to solving systems of linear equations. Direct methods, such as Gaussian elimination or LU decomposition, compute the exact solution in a finite number of steps but can be computationally expensive and memory-intensive for large systems. In contrast, iterative methods, like the Jacobi method or conjugate gradient, approximate the solution iteratively, refining an initial guess. While they do not guarantee an exact solution in a finite number of iterations, they are often more efficient in terms of computational time and memory usage, especially for sparse systems. The choice between a direct and iterative method depends on the size of the system, the structure of the matrix, the desired accuracy, and the available computational resources.

Definition 1:

An iterative method for solving the linear system Ax=b is a systematic approach that generates a sequence $X^{(k)}\ where\ k \in IN$. Each iterate $X^{(k)}$ is computed based on the previous iterates $X^{(0)}$,…, $X^{(k-1)}$, with the goal of converging to the solution X of the linear system. Typically, the construction of the sequence follows a recurrence relation of the form shown in Equation (23):

$$X^{(k)} = \boldsymbol{B}X^{(k-1)} + C \tag{23}$$

In this equation, $\boldsymbol{B}$ represents the iteration matrix derived from A, and C is a vector that depends on b. This formulation allows for flexibility in how the iterates are generated. By examining the limit as k approaches infinity ($k \to +\infty$), we observe that the solution X must satisfy Eq.(24):

$$X = BX + C \tag{24}$$

This reveals an important relationship between the solution and the matrices involved. Since we know that $X = A^{-1}b$, it follows that C can be expressed in Eq. (25) as:

$$C = (Id - b)\ A^{-1}b \tag{25}$$

This leads to the conclusion that the iterative method is fundamentally defined by the iteration matrix $\boldsymbol{B}$. Thus, understanding the structure and properties of $\boldsymbol{B}$ is crucial for analyzing the convergence and efficiency of the iterative method in solving the linear system $AX = b$.

Definition 2:

Splitting, A general technique for constructing the matrix $\boldsymbol{B}$ is based on a decomposition (or splitting) of the matrix A in the form:

$$A = P - N \tag{26}$$

Where P is an invertible matrix that is easy to invert, such as a diagonal or triangular matrix. The matrix P is referred to as the conditioning matrix. This approach is fundamental in iterative methods because it simplifies the process of finding solutions to linear systems.

$$PX^{(k+1)} = NX^{(k)} + b \;\rightarrow\; X^{(k+1)} = P^{-1}NX^{(k)} + P^{-1}b$$
$$where\ B = P^{-1}N\ and\ \ C = P^{-1}b \tag{27}$$

In this context, the matrix P plays a critical role as it dictates the stability and convergence properties of the iterative algorithm. The idea is to isolate the more easily manageable part of the matrix A (represented by P) from the more complex part (represented by N). By rearranging A in this way, we can define an iterative algorithm that can be expressed in a recurrence relation, allowing us to compute the next iterate based on the previous one.

The iterative algorithm can then be written as Eq. '23) ($X^{(k)} = \boldsymbol{B}X^{(k-1)} + C$).

where $\boldsymbol{B}$ is derived from the decomposition, and C incorporates the vector b. This formulation allows us to systematically update our estimates of the solution X in each iteration. The flexibility of choosing P allows for various strategies to optimize convergence, depending on the characteristics of the specific linear system being solved.

Overall, the splitting technique is a powerful tool in numerical linear algebra, providing a structured method to analyze and implement iterative methods effectively.

In practice, the most common splittings are based on the representation:

$$A = D - E - F \tag{28}$$

$$D = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} \qquad -E = \begin{pmatrix} 0 & 0 & \dots 0 \\ a_{21} & 0 & \dots 0 \\ \vdots & \vdots & \vdots \\ a_{n2} & a_{n2} & \cdots 0 \end{pmatrix} \qquad -F = \begin{pmatrix} 0 & a_{12} \dots a_{1n} \\ 0 & 0 \quad \dots a_{2n} \\ \vdots & \vdots \qquad \vdots \\ 0 & 0 \quad \cdots \ 0 \end{pmatrix}$$

Where D is the diagonal part of A, E is the strictly lower triangular part (with zeros on the diagonal), and F is the strictly upper triangular part (also with zeros on the diagonal) as shown in Eq. (28).

3.2 Jacobi and Relaxation Methods

3.2.1 Jacobi Method

The Jacobi method is an iterative technique for solving systems of linear equations of the form $AX = b$. This method is particularly useful for large and sparse matrices, where direct methods may be inefficient. There are two primary approaches to implementing the Jacobi method: the classical format based on explicit equations and the alternative format utilizing matrix splitting.

Both approaches have their advantages and applications, and understanding them provides a comprehensive view of how the Jacobi method operates in practice. In this course, we will delve into both methods, starting with the classical formulation before exploring the benefits of the splitting technique.

3.2.1.1 Equations-Based Formula

To implement the corresponding algorithm, consider the following system of linear equations in Eq. (29). This method involves directly manipulating the equations to derive the iterative formula.

$$AX = b \rightarrow \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{29}$$

Therefore, by isolating x_1 from the first equation, x_2 from the second equation, ..., and x_n from the n^{th} equation, we obtain the expression given by Eq. (30). Each variable is isolated and expressed in terms of the other variables, leading to a straightforward computation in each iteration.

$$\begin{cases} x_1 = \frac{(b_1 - a_{12}x_2 - \cdots - a_{1n}x_n)}{a_{11}} \\ x_2 = \frac{(b_2 - a_{21}x_1 - \cdots - a_{2n}x_n)}{a_{22}} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ x_n = \frac{(b_n - a_{n1}x_1 - \cdots - a_{n(n-1)}x_{n-1})}{a_{nn}} \end{cases} \rightarrow \forall i = 1..n, x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{ij}x_i\right) \tag{30}$$

Derivation of the Iterative Formula

Starting from a given initial vector $X^{(0)}$, we construct the sequence of vectors $(X^{(k)})_{k \geq 0}$, defined by Eq. (31).

$$\forall i = 1 \dots n,\ x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{ij}x_i^{(k)}\right) \tag{31}$$

Thus, by juxtaposing Eq. (31) with Eq. (23): ($X^{(k)} = \boldsymbol{B}X^{(k-1)} + C$), we obtain the iteration matrix $\boldsymbol{B_j}$ for the Jacobi method and a vector C in Eq. (32)

$$\boldsymbol{B_j} = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \cdots & -\frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & & \vdots \\ -\frac{a_{1n}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \cdots & 0 \end{pmatrix} \quad and \quad C = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{pmatrix} \tag{32}$$

3.2.1.2 Splitting-Based Formula

This technique involves decomposing the matrix A into its components: specifically the diagonal part D, the strictly lower triangular part E, and the strictly upper triangular part F. This decomposition simplifies the iteration process and enhances the clarity of the algorithm, as shown in Eq. (28).

Derivation of the Iterative Formula

Given the equation $AX = b$, we can rewrite it using our decomposition:

$$DX = b + (E + F)X \tag{33}$$

Rearranging this leads to Eq. '34):

$$X = D^{-1}(b + (E + F)X) \tag{34}$$

To derive the iterative formula, we isolate X in Eq. (35).

$$X^{(k)} = D^{-1}(b + (E + F)X^{(k-1)}) \tag{35}$$

This formula indicates that the new iterate $X^{(k)}$ is computed based on the previous iterate $X^{(k-1)}$and the contributions from the off-diagonal elements, represented by E and F.

By substituting Eq. (35) into Eq. (23), we obtain the iteration matrix for the Jacobi method, as given in Eq. (36).

$$\boldsymbol{B_j} = D^{-1}(E+F) \qquad and \qquad C = D^{-1}b \tag{36}$$

3.2.1.3 Iterative Algorithm Steps

The iterative process of the Jacobi method begins with the initialization step, where an initial guess for the solution is selected. This initial guess is often a zero vector, although any reasonable approximation can be used. The choice of initial values can influence the speed of convergence, but the method is generally robust enough to work with various starting points.
Once the initial guess is established, the process moves into the iteration phase. For each iteration, a new estimate is calculated based on the previous iterate. This step refines the approximation of the solution, incorporating the contributions from other variables in the system.
After computing the new estimate, it is essential to perform a convergence check to determine if the iterative process should continue. This is typically done by evaluating a stopping criterion, such as checking whether the difference between consecutive estimates is less than a predetermined small tolerance value. If this difference is sufficiently small, it indicates that the solution has stabilized, and the method can be terminated. This systematic approach ensures that the Jacobi method converges efficiently toward the true solution of the linear system.

1. **Initialization**: Choose an initial guess $X^{(0)}$ (often a zero vector).
2. **Iteration**: For each iteration k:

Calculate $X^{(k)}$using the formula

$$X^{(k)} = \boldsymbol{B_j}X^{(k-1)} + C$$

3. **Convergence Check**: Determine if the method has converged by evaluating the stopping criterion, such as:

$$\left\|X^{(k)} - X^{(k-1)}\right\| < \varepsilon$$

where ε epsilonϵ is a small tolerance value.

Example:

Consider the following system of equations:

$$\begin{cases} 3x_1 + x_2 - x_3 = 2 \\ x_1 + 5x_2 + 2x_3 = 17 \\ 2x_1 - x_2 \pm 6x_3 = -18 \end{cases}$$

1) **Equations-Based Formula**

$$\begin{cases} x_1 = \dfrac{(b_1 - a_{12}x_2 - a_{13}x_3)}{a_{11}} \\ x_2 = \dfrac{(b_2 - a_{21}x_1 - a_{23}x_3)}{a_{22}} \\ x_3 = \dfrac{(b_3 - a_{31}x_1 - a_{32}x_2)}{a_{11}} \end{cases} \rightarrow \begin{cases} x_1 = \dfrac{(2 - x_2 + x_3)}{3} \\ x_2 = \dfrac{(17 - x_1 - 2x_3)}{5} \\ x_3 = \dfrac{(-18 - 2x_1 + x_2)}{6} \end{cases}$$

Iterative Formula based on equation

$$\begin{cases} {x_1}^{(k+1)} = \dfrac{\left(2 - {x_2}^{(k)} + {x_3}^{(k)}\right)}{3} \\ {x_2}^{(k+1)} = \dfrac{\left(17 - {x_1}^{(k)} - 2{x_3}^{(k)}\right)}{5} \\ {x_3}^{(k+1)} = \dfrac{(-18 - 2{x_1}^{(k)} + {x_2}^{(k)})}{6} \end{cases}$$

$$\boldsymbol{B_j} = \begin{pmatrix} 0 & -1/3 & 1/3 \\ -1/5 & 0 & -2/5 \\ -1/3 & 1/6 & 0 \end{pmatrix} \quad and \quad C = \begin{pmatrix} 2/3 \\ 17/5 \\ -3 \end{pmatrix}$$

The iterative process of the Jacobi method:

If $X^{(0)} = \begin{pmatrix} \frac{2}{3} \\ \frac{17}{5} \\ -3 \end{pmatrix} \rightarrow \begin{cases} {x_1}^{(1)} = \frac{(2-{x_2}^{(0)}+{x_3}^{(0)})}{3} \\ {x_2}^{(1)} = \frac{(17-{x_1}^{(0)}-2{x_3}^{(0)})}{5} \\ {x_3}^{(1)} = \frac{(-18-2{x_1}^{(0)}+{x_2}^{(0)})}{6} \end{cases} \rightarrow \begin{cases} {x_1}^{(1)} = \frac{\left(2-(\frac{17}{5})+(-3)\right)}{3} = 0{,}5333 \\ {x_2}^{(1)} = \frac{\left(17-(\frac{2}{3})-2(-3)\right)}{5} = 2{,}0666 \\ {x_3}^{(1)} = \frac{(-18-2(\frac{2}{3})+(\frac{17}{5}))}{6} = 2{,}6555 \end{cases}$

NB: The result is computed with 4 significant digits (or 3 exact decimal places). So, probably, ε =0,5.10^{-3}.

$$\left\|X^{(1)} - X^{(0)}\right\| < 0.5.10^{-3} \rightarrow \left\|X^{(1)} - X^{(0)}\right\| = \left\|\begin{matrix} 0.5333 - \frac{2}{3} \\ 2.0666 - \frac{17}{5} \\ 2.6555 - 3 \end{matrix}\right\| = \left\|\begin{matrix} 0.5333 - \frac{2}{3} \\ 2.0666 - \frac{17}{5} \\ 2.6555 - 3 \end{matrix}\right\| = \left\|\begin{matrix} 1.1999 \\ 1.3334 \\ 0.3445 \end{matrix}\right\|$$

$\left\|\begin{matrix} 1.1999 \\ 1.3334 \\ 0.3445 \end{matrix}\right\| > \begin{pmatrix} 0.5.10^{-3} \\ 0.5.10^{-3} \\ 0.5.10^{-3} \end{pmatrix}$ So we should continue to process and calculate $X^{(1)}$.

2) **Splitting-Based Formula**

In our Sytem AX=b:

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & 5 & 2 \\ 2 & -1 & -6 \end{pmatrix} \quad and \quad b = \begin{pmatrix} 2 \\ 17 \\ -18 \end{pmatrix}$$

A=D-E-F

$$D=\begin{pmatrix}3&0&0\\0&5&0\\0&0&-6\end{pmatrix}\qquad E=\begin{pmatrix}0&0&0\\-1&0&0\\-2&1&0\end{pmatrix}\qquad F=\begin{pmatrix}0&-1&1\\0&0&-2\\0&0&0\end{pmatrix}$$

Calculate $\boldsymbol{B_j}=D^{-1}(E+F)\ and\ \ C=D^{-1}b$

$$D^{-1}=\begin{pmatrix}1/3&0&0\\0&1/5&0\\0&0&-1/6\end{pmatrix}$$ Calculate

$Calculate\ D^{-1}using\ Gauss\ jordan\ method\ or\ D^{-1}=\frac{1}{\det\ (D)}\times adj(D)$

$$(E+F)=\begin{pmatrix}0&-1&1\\-1&0&-2\\-2&1&0\end{pmatrix}$$

$$\boldsymbol{B_j}=D^{-1}(E+F)\ \rightarrow\boldsymbol{B_j}=\begin{pmatrix}\frac{1}{3}&0&0\\0&\frac{1}{5}&0\\0&0&-\frac{1}{6}\end{pmatrix}\begin{pmatrix}0&-1&1\\-1&0&-2\\-2&1&0\end{pmatrix}=\begin{pmatrix}0&-1/3&1/3\\-1/5&0&-2/5\\-1/3&1/6&0\end{pmatrix}$$

$$C=D^{-1}b=\begin{pmatrix}\frac{1}{3}&0&0\\0&\frac{1}{5}&0\\0&0&-\frac{1}{6}\end{pmatrix}\begin{pmatrix}2\\17\\-18\end{pmatrix}=\begin{pmatrix}2/3\\17/5\\-3\end{pmatrix}$$

$$\boldsymbol{B_j}=\begin{pmatrix}0&-1/3&1/3\\-1/5&0&-2/5\\-1/3&1/6&0\end{pmatrix}\quad\text{and}\quad C=\begin{pmatrix}2/3\\17/5\\-3\end{pmatrix}$$

Iterative Formula based on matrix:

$$\boldsymbol{X}^{(\boldsymbol{k}+\boldsymbol{1})}=\begin{pmatrix}0&-1/3&1/3\\-1/5&0&-2/5\\-1/3&1/6&0\end{pmatrix}\boldsymbol{X}^{(\boldsymbol{k})}+\begin{pmatrix}2/3\\17/5\\-3\end{pmatrix}$$

Exercise: Consider the following matrix

$$A=\begin{pmatrix}4&-1&0\\-1&2&0\\0&0&3\end{pmatrix}\ and\ B=\begin{pmatrix}2\\5\\1\end{pmatrix}$$

How many iterations for jacobi method needed to get an accuracy within 10^{-2}.

3.2.2 Relaxation Method

The relaxation method refers to a general family of iterative techniques that gradually refine the solution to a system of equations. It involves adjusting the current estimate of the solution by combining it with some computed corrections.

The idea is to improve convergence by introducing a relaxation factor (usually denoted as ω). In its simplest form, the relaxation method updates the solution by:

$$\begin{cases} X^{(0)} \in IR^n \\ \qquad x_i^{(k+1)} = x_i^{(k)} + \omega\left(\frac{b_i - -\sum_{j>i} a_{ij} x_j^{(k)}}{a_{ii}}\right) \end{cases} \qquad (37)$$

The relaxation parameter, usually denoted as ω\omegaω, determines the step size of the update. It lies in the range $0 < \omega < 1$

3.3 Gauss-Seidel and Successive Relaxation Methods

3.3.1 Gauss-Seidel Method

The Gauss-Seidel method is a widely-used iterative technique for solving systems of linear equations, particularly advantageous for large-scale and sparse matrices. One of its key benefits is its simplicity, making it accessible for a variety of applications. Additionally, it often converges faster than the Jacobi method, providing quicker solutions, especially in scenarios involving sparse matrices where many elements are zero, thus minimizing computational overhead. However, while generally reliable, the method's convergence is not guaranteed for all systems, and it can be sensitive to the choice of initial guess. Understanding these advantages and limitations helps in effectively utilizing Gauss-Seidel, especially in conjunction with techniques like Successive Over-Relaxation and in parallel computing environments.

Typically, the construction of the detailed equation sequence follows a recurrence relation of the form shown in Equation (38):

$$\begin{cases} X^{(0)} \in IR^n \\ x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\, x_j^{(k+1)} - \sum_{j>i} a_{ij}\, x_j^{(k)}\right) \end{cases} \qquad (38)$$

In this equation, $x_i^{(k+1)}$ represents the updated value of the variable $x_i$in the $(k+1)^{th}$iteration. The term $b_i$is the corresponding constant from the system of equations, while a_{ij} are the coefficients from the matrix A. The first summation accounts for the contributions of already updated variables, and the second summation includes the contributions from previous iteration values, ensuring that the most current data is utilized. This iterative approach allows for progressive refinement of the solution, showcasing the method's efficiency, particularly in handling large, sparse systems.

Using this approach, we can represent the iterative update in Eq. (27) as: $X^{(k+1)} = P^{-1}NX^{(k)} + P^{-1}b \ \ where\ B = P^{-1}N\ and\ \ C = P^{-1}b$. It consists of choosing the simplest splitting with P = D-E et N = F. By replacing in the equation Eq. (27), we obtain Eq. (39).

$$X^{(k+1)} = (D-E)^{-1}FX^{(k)} + (D-E)^{-1}b$$

$$\boldsymbol{B_{GS}} = (D-E)^{-1}F \qquad and \qquad C = (D-E)^{-1}b \tag{39}$$

Example:

Consider the same system of equations:

$$\begin{cases} 3x_1 + x_2 - x_3 = 2 \\ x_1 + 5x_2 + 2x_3 = 17 \\ 2x_1 - x_2 \pm 6x_3 = -18 \end{cases}$$

Based on A=D-E-F

$$D = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -6 \end{pmatrix} \qquad E = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -2 & 1 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix}$$

Calculate $\boldsymbol{B_{SOR}} = (D-E)^{-1}\left(\left(\frac{1-\omega}{\omega}\right)D + F\right) \qquad and \qquad C = (D-E)^{-1}b$

$$(D-E) = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -6 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 1 & 5 & 0 \\ 2 & -1 & -6 \end{pmatrix}$$

$$(D-E)^{-1} = \begin{pmatrix} 0.333 & 0 & 0 \\ -0.066 & 0.2 & 0 \\ 0.122 & -0.033 & -0.166 \end{pmatrix}$$

$Calculate\ (D-E)^{-1}\ \ using\ Gauss\ jordan\ method\ or\ (D-E)^{-1} = \frac{1}{\det\,(D-E)} \times adj((D-E))$

$$\boldsymbol{B_j} = (D-E)^{-1}F \ \to \boldsymbol{B_j} = \begin{pmatrix} 0.333 & 0 & 0 \\ -0.066 & 0.2 & 0 \\ 0.122 & -0.033 & -0.166 \end{pmatrix} \begin{pmatrix} 0. & -1 & 1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & -0.333 & 0.333 \\ 0 & 0.066 & -0.466 \\ 0 & -0.122 & 0.188 \end{pmatrix}$$

$$C = (D-E)^{-1}b = \begin{pmatrix} 0.333 & 0 & 0 \\ -0.066 & 0.2 & 0 \\ 0.122 & -0.033 & -0.166 \end{pmatrix} \begin{pmatrix} 2 \\ 17 \\ -18 \end{pmatrix} = \begin{pmatrix} 0.666 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$\boldsymbol{B_j} = \begin{pmatrix} 0 & -0.333 & 0.333 \\ 0 & 0.066 & -0.466 \\ 0 & -0.122 & 0.188 \end{pmatrix}$ and C=$\begin{pmatrix} 0.666 \\ 3.208 \\ 2.671 \end{pmatrix}$

Gauss Seidal Iterative Formula based on matrix:

$$\boldsymbol{X^{(k+1)}} = \begin{pmatrix} 0 & -0.333 & 0.333 \\ 0 & 0.066 & -0.466 \\ 0 & -0.122 & 0.188 \end{pmatrix} \boldsymbol{X^{(k)}} + \begin{pmatrix} 0.666 \\ 3.208 \\ 2.671 \end{pmatrix}$$

3.3.2 Successive Relaxation Method (SOR)

Relaxation methods are iterative techniques used to solve systems of linear equations, particularly in numerical analysis and computational mathematics. In this method, we slightly modify the previous method by introducing a parameter w, the relaxation coefficient. This parameter is generally constant. Relaxation in the Jacobi method typically does not provide any significant gains (see sub-section 3.2.2. However, when applied to the Gauss-Seidel method (see sub-section 3.3.1), it improves the speed of convergence. The update formula becomes:

$$\begin{cases} X^{(0)} \in IR^n \\ x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}(b_i - \sum_{j<i} a_{ij}\, x_j^{(k+1)} - \sum_{j>i} a_{ij}\, x_j^{(k)}) \\ \end{cases} \tag{40}$$

In Eq. (40):

- $x_i^{(k+1)}$ is the updated value for the i^{th} variable.
- $x_j^{(k+1)}$ are the most recently updated values for indices $j < i$ and $x_i^{(k)}$ are the previous values for indices $j > i$.

The idea is that if the "*correction*" applied to a component is going in the "right direction," we benefit from increasing it by multiplying by a factor greater than 1 (ω>1\omega > 1ω>1: over-relaxation). Conversely, if there is a risk of diverging or oscillating, it is better to dampen the correction by multiplying by a factor less than 1 (ω<1\omega < 1ω<1: under-relaxation). A necessary but not sufficient condition for the convergence of these methods is that the parameter ω lies between 0 and 2.

Given the equation $AX = b$ we can rewrite it using our decomposition into the components D , E, and F $(\,where\ A = D - E - F\ see\ Section$ 3.1):

$$X^{(k+1)} = \left(\frac{D}{\omega} - E\right)^{-1}\left(\left(\frac{1-\omega}{\omega}\right)D + F\right)X^{(k)} + \left(\frac{D}{\omega} - E\right)^{-1} b \tag{41}$$

By substituting Eq. (41) into Eq. (23), we obtain the iteration matrix for the SOR method, as given in Eq. (42).

$$\boldsymbol{B_{SOR}} = \left(\frac{D}{\omega} - E\right)^{-1}\left(\left(\frac{1-\omega}{\omega}\right)D + F\right) \qquad and \qquad C = \left(\frac{D}{\omega} - E\right)^{-1} b \tag{42}$$

The iterative process for the SOR method will remain the same as described in subsection 3.2.1.3.

Example:

Consider the following system of equations and $= 1.1$:

$$\begin{cases} 3x_1 + x_2 - x_3 = 2 \\ x_1 + 5x_2 + 2x_3 = 17 \\ 2x_1 - x_2 \pm 6x_3 = -18 \end{cases}$$

Based on A=D-E-F

$$D = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -6 \end{pmatrix} \qquad E = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -2 & 1 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix}$$

Calculate $\boldsymbol{B_{SOR}} = \left(\frac{D}{\omega} - E\right)^{-1}\left(\left(\frac{1-\omega}{\omega}\right)D + F\right) \quad and \quad C = \left(\frac{D}{\omega} - E\right)^{-1} b$

$$\left(\frac{D}{\omega} - E\right) = \begin{pmatrix} \frac{3}{1.1} & 0 & 0 \\ 0 & \frac{5}{1.1} & 0 \\ 0 & 0 & -\frac{6}{1.1} \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2{,}72 & 0 & 0 \\ 1 & 4{,}54 & 0 \\ 2 & -1 & -5{,}45 \end{pmatrix}$$

$$\left(\frac{D}{\omega} - E\right)^{-1} = \begin{pmatrix} 0.37 & 0 & 0 \\ -0.08 & 0.22 & 0 \\ 0.15 & -0.04 & -0.18 \end{pmatrix}$$

$Calculate \left(\frac{D}{\omega} - E\right)^{-1} \quad using\ Gauss\ jordan\ method\ or \left(\frac{D}{\omega} - E\right)^{-1} = \frac{1}{\det\left(\frac{D}{\omega} - E\right)} \times adj(\left(\frac{D}{\omega} - E\right))$

$$\left(\left(\frac{1-\omega}{\omega}\right)D + F\right) = \left(\left(\frac{1-1.1}{1.1}\right)D + F\right) = \left((-0{,}09)D + F\right) = \begin{pmatrix} -0.27. & -1 & 1 \\ 0 & -0.45 & -2 \\ 0 & 0 & 0.54 \end{pmatrix}$$

$$\boldsymbol{B_j} = \left(\frac{D}{\omega} - E\right)^{-1}\left(\left(\frac{1-\omega}{\omega}\right)D + F\right) \to \boldsymbol{B_j} = \begin{pmatrix} 0.37 & 0 & 0 \\ -0.08 & 0.22 & 0 \\ 0.15 & -0.04 & -0.18 \end{pmatrix}\begin{pmatrix} -0.27. & -1 & 1 \\ 0 & -0.45 & -2 \\ 0 & 0 & 0.54 \end{pmatrix}$$
$$= \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix}$$

$$C = \left(\frac{D}{\omega} - E\right)^{-1} b = \begin{pmatrix} 0.37 & 0 & 0 \\ -0.08 & 0.22 & 0 \\ 0.15 & -0.04 & -0.18 \end{pmatrix}\begin{pmatrix} 2 \\ 17 \\ -18 \end{pmatrix} = \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$\boldsymbol{B_j} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix}$ and C=$\begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$

Iterative Formula based on matrix:

$$\boldsymbol{X^{(k+1)}} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \boldsymbol{X^{(k)}} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$\boldsymbol{If\ X^{(0)}} = \begin{pmatrix} 0.00 \\ 0.00 \\ 0.00 \end{pmatrix}$ and In order to calculate the solution X using the Successive Over-Relaxation (SOR) method up to the 5th iteration, I will perform the following steps:

$$\boldsymbol{X}^{(1)} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \boldsymbol{X}^{(0)} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$$= \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \begin{pmatrix} 0.00 \\ 0.00 \\ 0.00 \end{pmatrix} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix} = \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix} \rightarrow \boldsymbol{X}^{(1)} = \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$$\boldsymbol{X}^{(2)} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \boldsymbol{X}^{(1)} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$$= \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix} = \begin{pmatrix} 0.407 \\ 2.18 \\ 2.73 \end{pmatrix} \rightarrow \boldsymbol{X}^{(2)} \approx \begin{pmatrix} 0.407 \\ 2.18 \\ 2.73 \end{pmatrix}$$

$$\boldsymbol{X}^{(3)} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \boldsymbol{X}^{(2)} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$$= \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \begin{pmatrix} 0.407 \\ 2.18 \\ 2.73 \end{pmatrix} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix} = \begin{pmatrix} 0.90 \\ 2.21 \\ 2.91 \end{pmatrix} \rightarrow \boldsymbol{X}^{(3)} \approx \begin{pmatrix} 0.90 \\ 2.21 \\ 2.91 \end{pmatrix}$$

$$\boldsymbol{X}^{(4)} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \boldsymbol{X}^{(3)} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$$= \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \begin{pmatrix} -2.14 \\ 3.96 \\ -5.33 \end{pmatrix} + \begin{pmatrix} 0.90 \\ 2.21 \\ 2.91 \end{pmatrix} = \begin{pmatrix} 0.92 \\ 2.12 \\ 2.91 \end{pmatrix} \rightarrow \boldsymbol{X}^{(4)} \approx \begin{pmatrix} 0.92 \\ 2.12 \\ 2.91 \end{pmatrix}$$

$$\boldsymbol{X}^{(5)} = \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \boldsymbol{X}^{(4)} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix}$$

$$= \begin{pmatrix} -0.09 & -0.37 & 0.37 \\ 0.02 & 0.02 & -0.52 \\ -0.04 & -0.13 & 0.13 \end{pmatrix} \begin{pmatrix} 0.92 \\ 2.12 \\ 2.91 \end{pmatrix} + \begin{pmatrix} 0.74 \\ 3.58 \\ 2.86 \end{pmatrix} = \begin{pmatrix} 0.94 \\ 2.12 \\ 2.92 \end{pmatrix} \rightarrow \boldsymbol{X}^{(5)} \approx \begin{pmatrix} 0.94 \\ 2.12 \\ 2.92 \end{pmatrix}$$

Thus, the result for the fifth iteration is:

$$\boldsymbol{X}^{(5)} \approx \begin{pmatrix} 0.94 \\ 2.12 \\ 2.92 \end{pmatrix} \; \textit{so the solution } X \rightarrow \begin{pmatrix} 1.00 \\ 2.00 \\ 3.00 \end{pmatrix}$$

3.4 Remarks on the Implementation of Iterative Methods

When implementing iterative methods for solving linear systems, several factors can significantly impact their efficiency and effectiveness. Here are some key considerations:

1. Initial Guess:

- **Choice:** The initial guess can significantly influence convergence speed. A good initial guess can accelerate convergence, while a poor one may lead to slow convergence or even divergence.
- **Strategies:** Common strategies include using a zero vector, averaging previous solutions, or leveraging prior knowledge about the system.

1) Zero Vector:

- **Example**: For a system $Ax = b$, starting with $x = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ is a common default. This can work well for many problems but may not be optimal for every system.

2) Averaging Previous Solutions:

This strategy can be effective in iterative algorithms where prior estimates are available. For instance, if previous solutions from a related problem or earlier iterations are known, averaging them can provide a more refined initial guess.

Example: If past solutions were $x^{(k-1)} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ and $x^{(k-2)} = \begin{pmatrix} 0.5 \\ 1.5 \\ 2.5 \end{pmatrix}$, an average can be $x^{(0)} = \frac{1}{2}(x^{(k-1)} + x^{(k-2)}) = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$.

3) Leveraging Prior Knowledge:

If there is existing knowledge about the system or the expected solution range, this can guide the choice of an initial guess. For example, in physical systems modeled by differential equations, parameters often lie within known bounds.

Example: If a temperature distribution in a rod is known to stabilize around a certain value based on physical properties, starting near that temperature can lead to faster convergence.

In summary, the initial guess in iterative methods is crucial for convergence speed and overall effectiveness. By employing strategies such as using a zero vector, averaging previous solutions, or leveraging prior knowledge, one can enhance the chances of rapid convergence to the desired solution.

2. Convergence Criteria:

Defining clear criteria for determining convergence is essential in iterative methods. Convergence criteria help establish when the solution has sufficiently approximated the true answer, allowing the algorithm to terminate.

Common Convergence Criteria

1) Difference Between Successive Iterates:

The most straightforward method involves comparing the difference between successive estimates of the solution. If the difference falls below a predefined tolerance level, the process can be considered converged.

Example: For a vector x, the criterion can be expressed as: $\left\|x^{(k+1)} - x^{(k)}\right\| < \varepsilon$ where ε is a small positive number (e.g., 10^{-6}). If the norm of the difference is smaller than ε, the algorithm stops.

2) **Residual Norm**:

Another common approach is to check the residual of the equation $Ax = b$. If the norm of the residual $r = b - Ax$ is sufficiently small, the solution is deemed converged.

Example: If $\|r\| < \delta$, where δ is another small tolerance (e.g., 10^{-6}), the solution is considered acceptable.

Adaptive Convergence Criteria

Adaptive criteria involve adjusting the convergence tolerance based on the current progress of the solution. This approach can improve efficiency by allowing the algorithm to adapt to the problem's dynamics.

Benefits of Adaptive Criteria

1) **Dynamic Adjustment**:

Instead of a fixed tolerance, the tolerance can be adjusted based on how quickly the solution is approaching convergence. For example, as the iterates get closer to the solution, the tolerance could become stricter.

Example: If the norm of the difference $\left\|x^{(k+1)} - x^{(k)}\right\|$ is decreasing rapidly, the algorithm could tighten the tolerance from 10^{-6} to 10^{-8}.

2) **Performance Improvement**:

Adaptive criteria can lead to faster convergence in some cases, especially for complex or ill-conditioned problems where a static tolerance might not be appropriate.

Example: In an optimization problem, if the function value decreases significantly in one iteration, the tolerance could be relaxed temporarily, allowing for faster exploration of the solution space.

Clearly defining convergence criteria is fundamental in iterative methods, ensuring that the solution process is both effective and efficient. By implementing adaptive criteria, one can tailor the convergence process to the specific dynamics of the problem, enhancing performance and reducing unnecessary computations.

3. Method Selection:

When selecting an iterative method for solving linear systems, the properties of the matrix play a crucial role in determining which algorithm will be most effective. Different methods have strengths and weaknesses depending on the characteristics of the matrix involved.

Key Matrix Properties

1) **Diagonally Dominant Matrices**:

A matrix A is diagonally dominant if for each row, the absolute value of the diagonal entry is greater than or equal to the sum of the absolute values of the other entries in that row.

Method Suitability: Methods like Gauss-Seidel and Jacobi perform well with diagonally dominant matrices due to their guaranteed convergence.

Example: For a matrix $A = \begin{pmatrix} 10 & 1 & -5 \\ 4 & -7 & 2 \\ 1 & 1 & 3 \end{pmatrix}$

each row satisfies the diagonal dominance condition, making it suitable for these methods.

2) **Symmetric Positive Definite Matrices**:

A matrix is symmetric positive definite if it is symmetric ($A = A^T$) and all its eigenvalues are positive.

Method Suitability: Iterative methods like Conjugate Gradient are specifically designed for symmetric positive definite matrices, providing efficient convergence.

Example: A matrix like $A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}$

is symmetric positive definite, making it ideal for methods like Conjugate Gradient.

3) **Sparse Matrices**:

Many large systems are represented by sparse matrices, which contain a significant number of zero elements.

Method Suitability: Iterative methods such as Krylov subspace methods (e.g., GMRES) are effective for sparse systems, as they can take advantage of the matrix's sparsity to reduce computational costs.

Example: A sparse matrix: $A = \begin{pmatrix} 0 & 0 & 1 \\ 2 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix}$ is well-suited for these methods.

Hybrid Approaches

Combining different iterative methods can leverage their strengths and mitigate their weaknesses, often resulting in improved convergence rates and more robust performance.

Examples of Hybrid Approaches

1) **Using Smoothing Techniques**:

   Combine a direct method (like LU decomposition) with an iterative method. For example, one might use LU to obtain a preliminary solution and then apply SOR to refine it.

   Example: Start with $X^{(0)}$from LU decomposition and then iteratively improve it using SOR.

2) **Multi-Grid Methods**:

   Multi-grid methods involve using different grid levels (coarse to fine) to accelerate convergence. They can effectively reduce errors at multiple scales.

   Example: In solving PDEs (Partial Differential Equation), one can use a coarse grid to solve for global features and then refine the solution on finer grids.

3) **Preconditioning**:

   Preconditioning involves transforming the original problem into a more favorable form before applying an iterative method. This can improve convergence rates significantly.

   Example: Applying an incomplete LU decomposition as a preconditioner for methods like Conjugate Gradient.

4) **Adaptive Strategy**:

   Employ an adaptive method that switches between different iterative techniques based on the convergence behavior observed during iterations.

   Example: Start with Jacobi for initial convergence and switch to Gauss-Seidel for refinement once closer to the solution.

Choosing an appropriate iterative method based on matrix properties is crucial for efficient problem-solving. Additionally, exploring hybrid approaches can capitalize on the strengths of various methods, leading to faster convergence and enhanced robustness. By understanding both the nature of the matrix and the available methods, one can optimize the solution process effectively.

4. Relaxation Parameters:

In iterative methods like Successive Over-Relaxation (SOR), the relaxation parameter ω\omegaω plays a crucial role in optimizing convergence.

1) **Tuning $\boldsymbol{\omega}$**

Careful tuning of ω can significantly impact convergence speed. For instance, a value around 1.25 often works well for diagonally dominant matrices, enhancing convergence compared to standard methods. Conversely, setting ω too high or too low can lead to divergence or excessively slow convergence. For example, using $\omega = 0.5$ may result in prolonged iterations, while $\omega = 3$ could cause oscillations.

2) Dynamic Adjustment of *ω*

Adapting ω during the iteration process can further enhance efficiency. As the solution approaches the true value, adjusting ω can help maintain optimal convergence. For example, starting with $\omega = 1.25$ and gradually reducing it to 1.1 as the iterates stabilize allows for fine-tuning. Implementing a feedback mechanism that monitors convergence rates and adjusts ω\omegaω accordingly can lead to better performance, such as increasing ω when improvements are sluggish.

By carefully tuning and dynamically adjusting the relaxation parameter ω, one can significantly improve the performance of iterative methods like SOR, ensuring faster and more reliable convergence.

5. Matrix Properties:

1) Condition Number

The condition number of a matrix significantly affects the convergence of iterative methods. A poorly conditioned matrix can lead to slow convergence and numerical instability. For example, a matrix with a high condition number (e.g., 1,000) may cause small changes in the input to produce large variations in the output, making it challenging to converge to an accurate solution.

2) Sparsity

Leveraging the sparsity of a matrix is crucial for optimizing computational cost and memory usage. Sparse matrices, which contain a significant number of zero elements, can be represented efficiently, reducing both storage requirements and computational complexity. For instance, using specialized algorithms like Conjugate Gradient can take advantage of matrix sparsity, leading to faster solutions for large-scale problems**.**

In summary, understanding the condition number is essential for anticipating convergence behavior, while utilizing sparsity can enhance the efficiency of iterative methods, ultimately leading to quicker and more resource-efficient computations.

6. Numerical Stability:

1) Rounding Errors

Rounding errors are a critical concern in iterative methods, as they can accumulate during calculations and impact the accuracy of the final solution. For example, in a sequence of iterative updates, small errors introduced at each step can compound, leading to significant deviations from the true solution. This is especially problematic in poorly conditioned matrices, where precision is crucial.

2) Preconditioning

Preconditioning techniques can significantly improve the condition number of a matrix, thereby accelerating convergence. By transforming the original system into a more favorable form, preconditioners make it easier for iterative methods to find a solution. For instance, applying an incomplete LU decomposition as a preconditioner can lead to faster convergence rates in methods like Conjugate Gradient, particularly for large, sparse systems.

In conclusion, being aware of rounding errors is essential for maintaining solution accuracy, while employing preconditioning techniques can enhance the efficiency of iterative methods by improving the condition number of the matrix and facilitating quicker convergence.

7. Parallelization:

Parallelizing iterative methods can significantly enhance performance by utilizing modern multi-core and distributed computing architectures. For example, in the Jacobi method, each element of the solution can be updated simultaneously, allowing for independent calculations across processors. This parallel approach is particularly beneficial for large-scale problems, where sequential processing would be inefficient.

Similarly, in methods like Conjugate Gradient, the computation of inner products and matrix-vector multiplications can be executed in parallel, leading to faster convergence and better resource utilization.

8. Implementation and Testing:

1) Language and Libraries

Selecting an appropriate programming language and leveraging optimized libraries is crucial for efficient implementation of iterative methods. For example, languages like Python or C++ offer powerful libraries such as NumPy and Eigen, respectively, which provide highly optimized functions for matrix operations. Utilizing these libraries can significantly enhance performance and reduce development time.

2) Testing

Thoroughly testing the implementation with various test cases is essential to ensure accuracy and identify potential issues. For instance, testing the algorithm on known solutions, edge cases, and larger problem sizes can help verify correctness and robustness. This process ensures that the method performs well across different scenarios and maintains stability under various conditions.

In summary, choosing the right programming language and libraries, along with rigorous testing, is vital for the effective implementation of iterative methods. These practices help optimize performance and ensure the reliability of the solution.

By carefully considering these factors and tailoring your implementation accordingly, you can enhance the performance and reliability of iterative methods for solving linear systems.

QCM

Here are multiple-choice questions (MCQs) based on previous section:

Question 1:

What is the impact of a good initial guess in iterative methods for solving linear systems?
A) It has no impact on convergence speed.
B) It can lead to slow convergence or divergence.
C) It can significantly accelerate convergence.
D) It only affects the final solution accuracy.
Answer: C) It can significantly accelerate convergence.

Question 2:

Which of the following is a common strategy for selecting an initial guess in iterative methods?
A) Using the identity matrix.
B) Averaging previous solutions.
C) Randomly generating values.
D) Always using zero as the initial guess.
Answer: B) Averaging previous solutions.

Question 3:

What does a matrix need to be for the Jacobi and Gauss-Seidel methods to guarantee convergence?
A) It must be sparse.
B) It must be symmetric.
C) It must be diagonally dominant or symmetric positive definite.
D) It must have a low condition number.
Answer: C) It must be diagonally dominant or symmetric positive definite.

Question 4:

How can the relaxation parameter ω\omegaω in Successive Over-Relaxation (SOR) be optimized?
A) It should always be set to 1.
B) It should be increased continuously throughout the iterations.
C) It can be tuned carefully to enhance convergence speed.
D) It is irrelevant to the convergence process.
Answer: C) It can be tuned carefully to enhance convergence speed.

Question 5:

What is the purpose of preconditioning techniques in iterative methods?
A) To increase the number of iterations needed for convergence.
B) To transform the problem into a more favorable form and improve the condition number of the matrix.
C) To simplify the matrix to a diagonal form.
D) To eliminate the need for convergence criteria.
Answer: B) To transform the problem into a more favorable form and improve the condition number of the matrix.

3.5 Convergence of Jacobi and Gauss-Seidel Methods

Definition 1

A square matrix $A \in IR_{nxn}$ is said to be ***diagonally dominant*** if, for each row i, the absolute value of the diagonal entry is greater than or equal to the sum of the absolute values of the other entries in that row. Mathematically, this can be expressed in Eq. (43) as:

$$\forall i = 1..n, |a_{ii}| \geq \sum_{j \neq i, j=1..n} |a_{ij}| \tag{43}$$

for all i If the inequality is strict for at least one row, the matrix is called SDD, ***strictly diagonally dominant***.

Example:

$$A = \begin{pmatrix} 5 & 1 & -1 \\ 2 & 3 & 0 \\ 3 & -1 & -7 \end{pmatrix} \rightarrow \begin{cases} |a_{11}| = 5 & > \quad |a_{12}| + |a_{13}| = 1 + 1 = 2 \\ |a_{22}| = 3 & > \quad |a_{21}| + |a_{23}| = 2 + 0 = 2 \\ |a_{33}| = 7 & > \quad |a_{31}| + |a_{32}| = 3 + 1 = 2 \end{cases} \rightarrow \quad So\ A\ is\ (SDD)$$

Definition 2

The matrix $A \in IR_{nxn}$is symmetric, meaning $A = A^T$ this implies that for $\forall i, j = 1..n, a_{ij} = a_{ji}$

Example:

$$A = \begin{pmatrix} 5 & 2 & -1 \\ 2 & 3 & 0 \\ -1 & 0 & -7 \end{pmatrix} \rightarrow A\ is\ symetric\ \forall i, j = 1..3, a_{ij} = a_{ji}$$

Definition 3

Let $A \in IR_{nxn}$ be a matrix. The principal minors of order k of this matrix are the determinants of the truncated matrices $\left(a_{ij}\right)_{1 \leq i, j \leq k}$ for $k$ranging from 1 to n.

$$A = \begin{pmatrix} a_{11} a_{12} \dots a_{1n} \\ a_{21} a_{22} \dots a_{2n} \\ \vdots \quad \vdots \quad\quad \vdots \\ a_{n2} a_{n2} \cdots\ a_{nn} \end{pmatrix} \rightarrow \begin{cases} D_1 = a_{11} \\ D_2 = det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \\ \quad\quad\quad \vdots \\ D_n = det \begin{pmatrix} a_{11} a_{12} \dots a_{1n} \\ a_{21} a_{22} \dots a_{2n} \\ \vdots \quad \vdots \quad\quad \vdots \\ a_{n2} a_{n2} \cdots\ a_{nn} \end{pmatrix} \end{cases} \rightarrow \forall i = 1..n, D_i > 0 \tag{44}$$

Where $D_i$are *Leading principal minors***.** These are the determinants of the square submatrices located in the upper left corner of A.

If all the leading principal minors of A are strictly positive, the matrix is said to be **positive definite**

Example:

$$A = \begin{pmatrix} 2 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \end{pmatrix}$$

$$\rightarrow \begin{cases} D_1 = a_{11} = 2 > 0 \\ D_2 = det\begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix} = \mathbf{-2 < 0} \\ \\ D_3 = det\begin{pmatrix} 2 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \end{pmatrix} = 2\begin{vmatrix} -1 & 1 \\ 0 & -2 \end{vmatrix} - 0\begin{vmatrix} 0 & 1 \\ 1 & -2 \end{vmatrix} + 12\begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix} = 4 + 1 = 5 > 0 \end{cases}$$

Since not all leading principal minors are positive. $D_2 = -2 < 0$, the matrix A is not positive definite.

Definition 4

Matrix norms $\|A\|$ provide a way to measure the size or magnitude of a matrix A. They are defined in terms of the elements of the matrix, denoted as a_{ij}.

1) **Max Norm (Infinity Norm)**:

$$\|A\|_\infty = max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| \qquad (45)$$

This norm takes the maximum absolute row sum of the matrix.

2) **1-Norm**:

$$\|A\|_1 = max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| \qquad (46)$$

This norm takes the maximum absolute column sum of the matrix.

Example:

$$A = \begin{pmatrix} 2 & 0 & -3 \\ 0 & -1 & 1 \\ 1 & 5 & -2 \end{pmatrix}$$

$$\|A\|_\infty = max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| = max_{1\le i\le 3}(|2| + |-3|, |-1| + |1|, |1| + |5| + |-2|)$$
$$= max_{1\le i\le 3}\begin{pmatrix} 5 \\ 2 \\ 7 \end{pmatrix} \rightarrow \|A\|_\infty = 7$$

$$\|A\|_1 = max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| = max_{1\le i\le 3}(|2| + |1|, |-1| + |5|, |-3| + |1| + |-2|)$$
$$= max_{1\le j\le 3}(3,6,6) \rightarrow \|A\|_1 = 6$$

Definition 5

The spectral radius of a matrix A is defined as the maximum absolute value of its eigenvalues λ_i. It is denoted in Eq. (47) as:

$$\rho(A) = max|\lambda_i| \tag{47}$$

where λ_i are the eigenvalues of the matrix A. The spectral radius provides important information about the stability and behavior of the matrix, particularly in applications related to dynamical systems and numerical analysis.

Example

Let's consider a simple 2x2 matrix:

$$A = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$$

To find the spectral radius, we first need to compute the eigenvalues of the matrix A.

1) Characteristic Polynomial: The eigenvalues are found by solving the characteristic equation given by: $det(A - \lambda I) = 0$.

   Where I is the identity matrix. For our matrix A:

   $$A - \lambda I = \begin{pmatrix} 4-\lambda & 2 \\ 1 & 3-\lambda \end{pmatrix}$$

2) Determinant Calculation**:**

   $$det(A - \lambda I) = (4-\lambda)(3-\lambda) - (2)(1) = \lambda^2 - 7\lambda + 10 = 0$$

3) Solving the Quadratic Equation:

   Using the quadratic formula $\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-7 \pm \sqrt{49-40}}{2} = \frac{-7 \pm 3}{2}$

   This gives us: $\lambda_1 = 5 \quad and \quad \lambda_2 = 2$

4) Calculating the Spectral Radius:

   Now, we compute the spectral radius: $\rho(A) = max|\lambda_i| = \rho(A) = \max(|5|, |2|) = 5$

3.5.1 Sufficient Conditions for Convergence of Iterative Methods

Iterative methods are widely used for solving linear systems, especially when dealing with large matrices. Among these methods, the Gauss-Seidel method and relaxation methods are particularly noteworthy. Their convergence properties can be influenced by the characteristics of the coefficient matrix A. In this section, we will discuss the sufficient conditions for the convergence of these methods, particularly focusing on the case when A is a strictly symmetric positive definite matrix.

3.5.1.1 Convergence of the Jacobi Method

The Jacobi method is another iterative approach used to solve the equation $AX = b$. Unlike the Gauss-Seidel method, which updates the solution using the latest values, the Jacobi method updates all components simultaneously using values from the previous iteration.

Sufficient Condition for Convergence: If A is strictly **symmetric positive definite**, the Jacobi method converges for any initial guess $X^{(0)}$. The convergence can be attributed to the following factors:

- Similar to the Gauss-Seidel method, the spectral radius $\rho(\boldsymbol{B_j})$ of the iteration matrix $\boldsymbol{B_j}$ associated with the Jacobi method is less than 1 ($\rho(\boldsymbol{B_j}) < 1$). This guarantees that the error decreases with each iteration.

3.5.1.2 Convergence of the Gauss-Seidel Method

The **Gauss-Seidel method** is an iterative approach that updates each component of x based on the most recent values, leveraging previously computed results.

Sufficient Condition for Convergence: If A is strictly symmetric positive definite, the Gauss-Seidel method converges for any initial guess $X^{(0)}$. This is due to:

- The spectral radius $\rho(\boldsymbol{B_{GS}})$ of the iteration matrix $\boldsymbol{B_{GS}}$ being less than 1 ($\rho(\boldsymbol{B_{GS}}) < 1$), ensuring that the error diminishes with each iteration.

3.5.1.3 Convergence of the Relaxation Method

The **relaxation method** is a generalization of the Gauss-Seidel method that introduces a relaxation factor ω.

Sufficient Condition for Convergence: For the relaxation method to converge, the relaxation factor ω must satisfy: $0<\omega<2$

When A is strictly symmetric positive definite, choosing ω\omegaω within the range (0,2) enhances the convergence speed. Specifically:

- When $\omega=1$, the method reduces to Gauss-Seidel.
- When $0<\omega<1$, the method may converge more slowly but still guarantees convergence.
- When $1<\omega<2$, the method often converges more quickly than standard Gauss-Seidel.

In summary, the conditions for convergence of the Jacobi method, Gauss-Seidel method, and the relaxation method are closely linked to the properties of the coefficient matrix A. Specifically, if A is strictly symmetric positive definite, all three methods will converge. For the relaxation method, selecting an appropriate relaxation factor ω (where $0<\omega<2$) is crucial for achieving optimal convergence rates. These insights are fundamental in numerical analysis and provide a strong basis for the effective implementation of iterative methods in practical applications.

Example:

To compute successive approximations of the solution of a system using the Jacobi and Gauss-Seidel methods, let's define a system of linear equations as follows:

$$A = \begin{pmatrix} a & b \\ b & a \end{pmatrix} \; and \;\; b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\begin{cases} ax + by = 1 \\ bx + ay = 1 \end{cases} \rightarrow \begin{cases} x^{k+1} = \dfrac{1}{a}(1 - by^k) \\ y^{k+1} = \dfrac{1}{a}(1 - bx^k) \end{cases}$$

Study of Convergence

1) Let's start with the **sufficient conditions**.

- $\forall i = 1..2, |a_{ii}| \geq \sum_{j \neq i, j=1..n} |a_{ij}| \rightarrow \begin{cases} |a| > |b| \\ |a| > |b| \end{cases}$

  the matrix A is said to be **(SDD) *strictly diagonally dominant*** if $|a| > |b|$

The Jacobi and Gauss-Seidel methods converges if $|a| > |b|$.

- $A = \begin{pmatrix} a & b \\ b & a \end{pmatrix} \; and \; A^T = \begin{pmatrix} a & b \\ b & a \end{pmatrix}$

  1) Since $A = A^T$, A is **symmetric**.
  2) the matrix A is said to be **positive definite** if all the leading principal minors of A are strictly positive $\forall i = 1..2, D_i > 0$.
    1. $D_1 = a_{11} = a > 0$
    2. $D_2 = \det A = \begin{vmatrix} a & b \\ b & a \end{vmatrix} = a^2 - b^2 > 0 \rightarrow |a| > |b|$

The Gauss-Seidel method converges if $|a| > |b|$.

2) Let's start with the **Necessary & Sufficient Conditions**

$$D = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}, E = \begin{pmatrix} 0 & 0 \\ -b & 0 \end{pmatrix}, F = \begin{pmatrix} 0 & -b \\ 0 & 0 \end{pmatrix}$$

Jacobi Method

$$\boldsymbol{B_j} = D^{-1}(E + F) \; and \;\; C = D^{-1}b$$

$$D^{-1} = \begin{pmatrix} 1/a & 0 \\ 0 & 1/a \end{pmatrix} \; and \; (E + F) = \begin{pmatrix} 0 & -b \\ -b & 0 \end{pmatrix}$$

$$\boldsymbol{B_j} = \begin{pmatrix} 0 & -b/a \\ -b/a & 0 \end{pmatrix} \; and \;\; C = \begin{pmatrix} 1/a \\ 1/a \end{pmatrix}$$

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

$$det(\,\boldsymbol{B_j} - \lambda I) = \det\begin{pmatrix} -\lambda & -\frac{b}{a} \\ -\frac{b}{a} & -\lambda \end{pmatrix} = \left(\lambda - \frac{b}{a}\right)\left(\lambda + \frac{b}{a}\right) = 0$$

$$det(\,\boldsymbol{B_j} - \lambda I) = 0 \rightarrow \lambda_1 = \frac{b}{a} \; and \; \lambda_2 = -\frac{b}{a}$$

The Jacobi method converges when $\rho(\boldsymbol{B_j}) = max|\lambda_i| < 1$

$$\rho(\boldsymbol{B_j}) = \max(|\lambda_1|, |\lambda_2|) = \left|\frac{b}{a}\right| < 1 \rightarrow |b| < |a|$$

The Jacobi method converges if $|a| > |b|$.

Gauss Seidel Method

$\boldsymbol{B_{GS}} = (D-E)^{-1}(F) \; and \;\; C = (D-E)^{-1}b$

$(D-E)^{-1} = \begin{pmatrix} 1/a & 0 \\ -b/a^2 & 1/a \end{pmatrix} \; and \; F = \begin{pmatrix} 0 & -b \\ 0 & 0 \end{pmatrix}$

$\boldsymbol{B_{GS}} = \begin{pmatrix} 0 & -b/a \\ 0 & b^2/a^2 \end{pmatrix}$

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

$$det(\,\boldsymbol{B_{GS}} - \lambda I) = \det\begin{pmatrix} -\lambda & -b/a \\ 0 & b^2/a^2 - \lambda \end{pmatrix} = (-\lambda)\left(\frac{b^2}{a^2} - \lambda\right) = 0$$

$$det(\,\boldsymbol{B_{GS}} - \lambda I) = 0 \rightarrow \lambda_1 = 0 \; and \; \lambda_2 = \frac{b^2}{a^2}$$

The Jacobi method converges when $\rho(\boldsymbol{B_{GS}}) = max|\lambda_i| < \left|\frac{b^2}{a^2}\right|$

$$\rho(\boldsymbol{B_{GS}}) = \max(|\lambda_1|, |\lambda_2|) = \left|\frac{b^2}{a^2}\right| < 1 \rightarrow |b| < |a|$$

The Gauss Seidel method converges if $|a| > |b|$.