Numerical Analysis Module 4
Solving Linear Algebraic Equations

Sachin C. Patwardhan
Dept. of Chemical Engineering,
Indian Institute of Technology, Bombay
Powai, Mumbai, 400 076, Inda.
Email: sachinp@iitb.ac.in

Contents

1 Introduction

The central problem of linear algebra is solution of linear equations of type

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 && (1)\\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 && (2)\\ ..... &= .... && (3)\\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &= b_m && (4) \end{aligned}$$

which can be expressed in vector notation as follows

$$\mathbf{Ax} = \mathbf{b} \tag{5}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & ... & ... & a_{mn} \end{bmatrix} \tag{6}$$

where $\mathbf{x} \in R^n, \mathbf{b} \in R^m$ and $A \in R^m \times R^n$ i.e. $\mathbf{A}$ is a $m \times n$ matrix. Here, m represents number of equations while n represents number of unknowns. Three possible situations arise while developing mathematical models

- **Case** $(m \geqslant n)$: system may have a unique solution / no solution / multiple solutions depending on rank of matrix $\mathbf{A}$ and vector $\mathbf{b}$.
- **Case** $(m < n)$: system either has no solution or has infinite solutions.

Note that when there is no solution, it is possible to find projection of $\mathbf{b}$ in the column space of $\mathbf{A}$. This case was dealt with in the Lecture Notes on Problem Discretization by Approximation Theory. In these lecture notes, we are interested in the case when $m = n$, particularly when the number of equations are large. A took-kit to solve equations of this type is at the heart of the numerical analysis. Before we present the numerical methods to solve equation (5), we discuss the conditions under which the solutions exist. We then proceed to develop direct and iterative methods for solving large scale problems. We later discuss *numerical conditioning* of a matrix and its relation to errors that can arise in computing numerical solutions.

2 Existence of Solutions

Consider the following system of equations

$$\begin{bmatrix} 1 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{7}$$

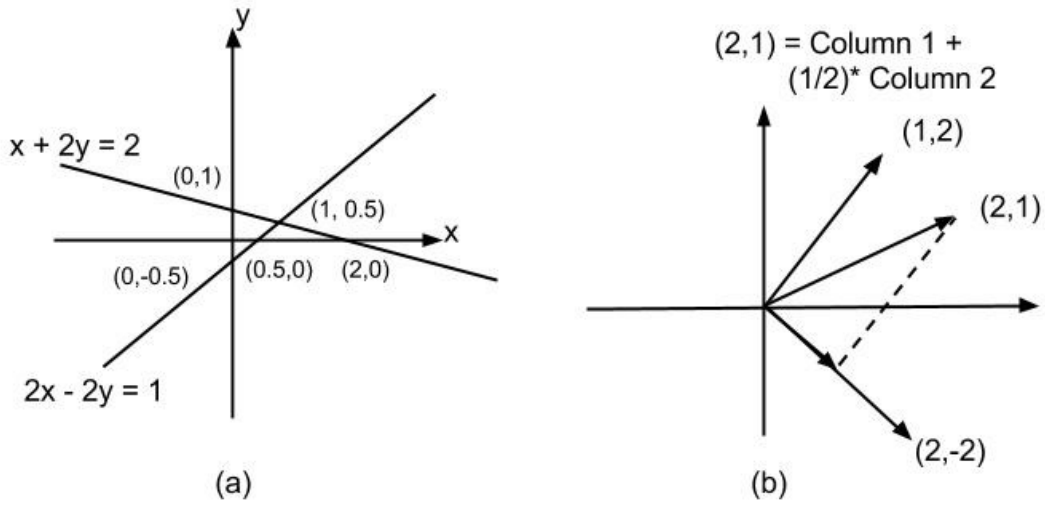

Figure 1: Linear algebraic equations: (a) Row viewpoint and (b) Column viewpoint

There are two ways of interpreting the above matrix vector equation geometrically.

- **Row viewpoint [3]:** If we consider two equations separately as

$$x + 2y = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 1 \tag{8}$$

$$2x - 2y = \begin{bmatrix} 2 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 1 \tag{9}$$

  then, each one is a line in x-y plane and solving this set of equations simultaneously can be interpreted as finding the point of their intersection (see Figure 1 (a)).

- **Column viewpoint** [3]: We can interpret the equation as linear combination of column vectors, i.e. as vector addition

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} x + \begin{bmatrix} 2 \\ -2 \end{bmatrix} y = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{10}$$

  Thus, the system of simultaneous equations can be looked upon as one vector equation i.e. addition or linear combination of two vectors (see Figure 1 (b)).

Now consider the following two linear equations

$$x + y = 2 \tag{11}$$

$$3x + 3y = 4 \tag{12}$$

This is clearly an inconsistent case and has no solution as the row vectors are linearly dependent. In column picture, no scalar multiple of

$$\mathbf{v} = [1\ 3]^T$$

can found such that $\alpha\mathbf{v} = [2\ 4]^T$. Thus, in a singular case

$$\text{Row viewpoint fails} \quad \Longleftrightarrow \quad \text{Column viewpoint fails}$$

i.e. if two lines fail to meet in the *row viewpoint* then vector $\mathbf{b}$ cannot be expressed as linear combination of column vectors in the *column viewpoint* [3].

Now, consider a general system of linear equations $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is an $n \times n$ matrix.

Row viewpoint : Let A be represented as

$$\mathbf{A} = \begin{bmatrix} \left(\mathbf{r}^{(1)}\right)^T \\ \left(\mathbf{r}^{(2)}\right)^T \\ .... \\ \left(\mathbf{r}^{(n)}\right)^T \end{bmatrix} \tag{13}$$

where $\left(\mathbf{r}^{(i)}\right)^T$ represents i'th row of matrix A. Then $\mathbf{Ax} = \mathbf{b}$ can be written as n equations

$$\begin{bmatrix} \left(\mathbf{r}^{(1)}\right)^T \mathbf{x} \\ \left(\mathbf{r}^{(2)}\right)^T \mathbf{x} \\ .... \\ \left(\mathbf{r}^{(n)}\right)^T \mathbf{x} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ .... \\ b_n \end{bmatrix} \tag{14}$$

Each of these equations $\left(\mathbf{r}^{(i)}\right)^T \mathbf{x} = b_i$ represents a hyperplane in R^n (i.e. line in R^2 ,plane in R^3 and so on). Solution of $\mathbf{Ax} = \mathbf{b}$ is the point $\mathbf{x}$ at which all these hyperplanes intersect (if at all they intersect in one point).

Column viewpoint : Let matrix $\mathbf{A}$ be represented as

$$\mathbf{A} = [\ \mathbf{c}^{(1)}\ \mathbf{c}^{(2)}.....\mathbf{c}^{(n)}\ ]$$

where $\mathbf{c}^{(i)}$ represents i^{th}column of $\mathbf{A}$. Then we can look at $\mathbf{Ax} = \mathbf{b}$ as one vector equation

$$x_1\mathbf{c}^{(1)} + x_2\mathbf{c}^{(2)} + ... + x_n\mathbf{c}^{(n)} = \mathbf{b} \tag{15}$$

Components of solution vector $\mathbf{x}$ tells us how to combine column vectors to obtain vector $\mathbf{b}$.In singular case, the n hyperplanes have no point in common or equivalently the n column vectors are not linearly independent. Thus, both these geometric interpretations are consistent with each other.

Matrix $\mathbf{A}$ operates on vector $\mathbf{x} \in R(\mathbf{A}^T)$ (i.e. a vector belonging to row space of $\mathbf{A}$) it produces a vector $\mathbf{Ax} \in R(\mathbf{A})$ (i.e. a vector in column space of $\mathbf{A}$). Thus, given a vector $\mathbf{b}$, a solution $\mathbf{x}$ exists $\mathbf{Ax} = \mathbf{b}$ if only if $\mathbf{b} \in R(\mathbf{A})$. The solution is unique if the columns of $\mathbf{A}$ are linearly independent and the null space of $\mathbf{A}$ contains only the origin**,** i.e. $\mathcal{N}(\mathbf{A}) \equiv \{\overline{\mathbf{0}}\}$. A non-zero null space is obtained only when columns of $\mathbf{A}$ are linearly dependent and the matrix $\mathbf{A}$ is not invertible. In such as case, we end up either with infinite number of solutions when $\mathbf{b} \in R(\mathbf{A})$ or with no solution when $\mathbf{b} \notin R(\mathbf{A})$.

3 Direct Solution Techniques

Methods for solving linear algebraic equations can be categorized as direct and iterative schemes. There are several methods which directly solve equation (5). Prominent among these are such as Cramer's rule, Gaussian elimination and QR factorization. As indicated later, the Cramer's rule is unsuitable for computer implementation and is not discussed here. Among the direct methods, we only present the Gaussian elimination here in detail.

3.1 Gaussian Elimination and LU Decomposition

The Gaussian elimination is arguably the most used method for solving a set of linear algebraic equations. It makes use of the fact that a solution of a special system of linear equations, namely the systems involving triangular matrices, can be constructed very easily. For example, consider a system of linear equation given by the following matrix equation

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & ... & u_{1n} \\ 0 & u_{22} & u_{23} & ... & u_{2n} \\ 0 & 0 & u_{33} & ... & u_{3n} \\ 0 & ... & .... & ... & ... \\ 0 & ... & . & 0 & u_{nn} \end{bmatrix} \tag{16}$$

Here, $\mathbf{U}$ is a upper triangular matrix such that all elements below the main diagonal are zero and all the diagonal elements are non-zero, i.e. $u_{ii} \neq 0$ for all i. To solve the system $\mathbf{Ux} = \boldsymbol{\beta}$, one can start from the last equation

$$x_n = \frac{\beta_n}{u_{nn}} \tag{17}$$

and then proceed as follows

$$\begin{aligned} x_{n-1} &= \frac{1}{u_{n-1,n-1}}\left[\beta_{n-1} - u_{n-1,n}x_n\right] \\ x_{n-2} &= \frac{1}{u_{n-2,n-2}}\left[\beta_{n-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n\right] \end{aligned}$$

In general, for i'th element x_i, we can write

$$x_i = \frac{1}{u_{ii}}\left[\beta_i - \sum_{j=i+1}^{n} u_{ij}x_j\right] \tag{18}$$

where $i = n-1, n-2, ...1$. Since we proceed from $i = n$ to $i = 1$, these set of calculations are referred to as *back substitution*.

Thus, the solution procedure for solving this system of equations involving a special type of upper triangular matrix is particularly simple. However, the trouble is that most of the problems encountered in real applications do not have such special form. Now, suppose we want to solve a system of equations $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a general full rank square matrix without any special form. Then, by some series of transformations, can we convert the system from $\mathbf{Ax} = \mathbf{b}$ form to $\mathbf{Ux} = \boldsymbol{\beta}$ form, i.e. to the desired special form? It is important to carry out this transformation such that the original system of equations and the transformed system of equations have identical solutions, $\mathbf{x}$.

To begin with, let us note one important property of the system of linear algebraic equations under consideration. Let $\mathbf{T}$ represent an arbitrary square matrix with full rank, i.e. $\mathbf{T}$ is invertible. Then, the system of equations $\mathbf{Ax} = \mathbf{b}$ and $(\mathbf{TA})\mathbf{x} = \mathbf{Tb}$ have identical solutions. This follows from invertibility of matrix $\mathbf{T}$. Thus, if we can find a matrix $\mathbf{T}$ such that $\mathbf{TA} = \mathbf{U}$, where $\mathbf{U}$ is of the form (16) and $u_{ii} \neq 0$ for all i, then recovering the solution $\mathbf{x}$ from the transformed system of equations is quite straight forward. Let us see how to construct such a transformation matrix systematically.

Thus, the task at hand is to convert a general matrix $\mathbf{A}$ to the upper triangular form $\mathbf{U}$.To understand the process of conversion, let is consider a specific example where $(\mathbf{A}, \mathbf{b})$ are chosen as follows

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 4 & 1 & 2 \\ -2 & 2 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 7 \\ 1 \end{bmatrix} \tag{19}$$

To begin with, we would like to transform $\mathbf{A}$ to matrix

$$\mathbf{A}^{(1)} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & * & * \\ -2 & 2 & 1 \end{bmatrix}$$

such that element (2,1) of $\mathbf{A}^{(1)}$ is zero. This can be achieved by constructing an *elementary matrix,* $\mathbf{E}_{21}$, as follows

$$\mathbf{E}_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -e_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$e_{21} = \frac{a_{21}}{a_{11}} = 2$$

Here, the element a_{11} is called as *pivot*, or, to be precise, the first pivot. Note that $\mathbf{E}_{21}$ is an invertible matrix and its inverse can be constructed as follows

$$(\mathbf{E}_{21})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ e_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying matrix $\mathbf{A}$ and vector $\mathbf{b}$ with $\mathbf{E}_{21}$ yields

$$\mathbf{A}^{(1)} = \mathbf{E}_{21}\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & -1 & 4 \\ -2 & 2 & 1 \end{bmatrix} \text{ and } \mathbf{b}^{(1)} = \mathbf{E}_{21}\mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

and since the transformation matrix $\mathbf{T} = \mathbf{E}_{21}$ is invertible, the transformed system of equations $(\mathbf{E}_{21}\mathbf{A})\mathbf{x} = (\mathbf{E}_{21}\mathbf{b})$ and the original system $\mathbf{Ax} = \mathbf{b}$ will have identical solutions. While the above transformation was achieved by multiplying both the sides of $\mathbf{Ax} = \mathbf{b}$ by $\mathbf{T} = \mathbf{E}_{21}$, identical result can be achieved in practice if we multiply the first row of the matrix augmented matrix $\left[\begin{array}{c|c} \mathbf{A} & \mathbf{b} \end{array}\right]$ by $-e_{21}$ add it to its second row, i.e.

$$a_{2j}^{(1)} = a_{2j} - e_{21}a_{1j}$$

for $j = 1, 2, ...n$ and $b_2^{(1)} = b_2 - e_{21}b_1$. This turns out to be much more efficient way of carrying out the transformation than doing the matrix multiplications.

Next step is to transform $\mathbf{A}^{(1)}$ to

$$\mathbf{A}^{(2)} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & -1 & 4 \\ 0 & * & * \end{bmatrix}$$

and this can be achieved by constructing

$$\mathbf{E}_{31} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -e_{31} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\dfrac{a_{31}}{a_{11}} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

and multiplying matrix $\mathbf{A}^{(1)}$ and vector $\mathbf{b}^{(1)}$ with $\mathbf{E}_{31}$ i.e.

$$\begin{aligned}\mathbf{A}^{(2)} &= \mathbf{E}_{31}\mathbf{A}^{(1)} = (\mathbf{E}_{31}\mathbf{E}_{21})\,\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & -1 & 4 \\ 0 & 3 & 0 \end{bmatrix} \\ \mathbf{b}^{(2)} &= \mathbf{E}_{31}\mathbf{b}^{(1)} = (\mathbf{E}_{31}\mathbf{E}_{21})\,\mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix}\end{aligned}$$

Note again that $\mathbf{E}_{31}$ is invertible and

$$(\mathbf{E}_{31})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ e_{31} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Thus, to begin with, we make all the elements in the first column zero, except the first one. To get an upper triangular form, we now have to eliminate element (3,2) of $\mathbf{A}^{(2)}$ and this can be achieved by constructing another elementary matrix

$$\mathbf{E}_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -e_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\dfrac{a_{32}^{(2)}}{a_{22}^{(2)}} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix}$$

Transforming $\mathbf{A}^{(2)}$ we obtain

$$\begin{aligned}\mathbf{U} &= \mathbf{A}^{(3)} = \mathbf{E}_{32}\mathbf{A}^{(2)} = (\mathbf{E}_{32}\mathbf{E}_{31}\mathbf{E}_{21})\,\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & -1 & 4 \\ 0 & 0 & 12 \end{bmatrix} \\ \boldsymbol{\beta} &= \mathbf{b}^{(3)}{=}\mathbf{E}_{31}\mathbf{b}^{(1)} = (\mathbf{E}_{32}\mathbf{E}_{31}\mathbf{E}_{21})\,\mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 12 \end{bmatrix}\end{aligned}$$

Note that matrix $\mathbf{A}^{(3)}$ is exactly the upper triangular form $\mathbf{U}$ that we have been looking for**.** The invertible transformation matrix $\mathbf{T}$ that achieves this is given by

$$\mathbf{T} = \mathbf{E}_{32}\mathbf{E}_{31}\mathbf{E}_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -5 & 3 & 1 \end{bmatrix}$$

which is a lower triangular matrix. Once we have transformed $\mathbf{Ax} = \mathbf{b}$ to $\mathbf{Ux} = \boldsymbol{\beta}$ form, it is straight forward to compute the solution

$$x_3 = 1,\ x_2 = 1 \text{ and } x_1 = 1$$

using the back substitution method.

It is interesting to note that

$$\mathbf{T}^{-1} = \mathbf{E}_{21}^{-1}\mathbf{E}_{31}^{-1}\mathbf{E}_{32}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ e_{21} & 1 & 0 \\ e_{31} & e_{32} & 1 \end{bmatrix}$$

i.e. the inverse of matrix $\mathbf{T}$ can be constructed simply by *inserting* e_{ij} at (i,j)'th position in an identity matrix. Since we have $\mathbf{U} = \mathbf{TA}$, matrix $\mathbf{A}$ can be recovered from $\mathbf{U}$ by inverting the transformation process as $\mathbf{A} = \mathbf{T}^{-1}\ \mathbf{U}$. Here, $\mathbf{T}^{-1}$ is a lower triangular matrix and it is customary to denote it using symbol $\mathbf{L}$ i.e. $\mathbf{L} \equiv \mathbf{T}^{-1}$ and $\mathbf{A} = \mathbf{L}\ \mathbf{U}$. This is nothing but the LU decomposition of matrix $\mathbf{A}$.

Before we proceed with generalization of the above example, it is necessary to examine a degenerate situation that can arise in the process of Gaussian elimination. In the course of transformation, we may end up with a scenario where the a zero appears in one of the pivot locations. For example, consider the system of equations with slightly modified $\mathbf{A}$ matrix and $\mathbf{b}$ vector i.e.

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 4 & 2 & 2 \\ -2 & 2 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 8 \\ 1 \end{bmatrix} \tag{20}$$

Multiplying with

$$\mathbf{E}_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{E}_{31} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

yields

$$\mathbf{A}^{(2)} = \mathbf{E}_{31}\mathbf{E}_{21}\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 0 & 4 \\ 0 & 3 & 0 \end{bmatrix} \text{ and } \mathbf{b}^{(2)} = \mathbf{E_{31}}\mathbf{E}_{21}\mathbf{b} = \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix}$$

Note that zero appears in the pivot element, i.e. $a_{22}^{(2)} = 0$, and consequently the construction of $\mathbf{E}_{32}$ is in trouble. This difficulty can be alleviated if we exchange the last two rows of matrix $\mathbf{A}^{(2)}$ and last two elements of $\mathbf{b}^{(2)}$, i.e.

$$\mathbf{A}^{(3)} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix} \text{ and } \mathbf{b}^{(3)} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

This rearrangement renders the pivot element $a_{22}^{(3)} \neq 0$ and the solution becomes tractable. The transformation leading to exchange of rows can be achieved by constructing a *permuta-*

tion matrix $\mathbf{P}_{32}$

$$\mathbf{P}_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

such that $\mathbf{P}_{32}\mathbf{A}^{(2)} = \mathbf{A}^{(3)}$. The permutation matrix $\mathbf{P}_{32}$ is obtained just by exchanging rows 2 and 3 of the identity matrix. Note that the permutation matrices have strange characteristics. Not only these matrices are invertible, but these matrices are inverses of themselves! Thus, we have

$$(\mathbf{P}_{32})^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which implies $(\mathbf{P}_{32})^{-1} = \mathbf{P}_{32}$ and, if you consider the fact that exchanging two rows of a matrix in succession will bring back the original matrix, then this result is obvious. Coming back to the construction of the transformation matrix for the linear system of equations under consideration, matrix $\mathbf{T}$ is now constructed as follows $\mathbf{T} = \mathbf{P}_{32}\mathbf{E}_{31}\mathbf{E}_{21}$.

This approach of constructing the transformation matrix $\mathbf{T}$, demonstrated using a 3×3 matrix $\mathbf{A}$, can be easily generalized for a system of equations involving an $n \times n$ matrix $\mathbf{A}$. For example, elementary matrix $\mathbf{E}_{i1}$, which reduces element $(i, 1)$ in matrix $\mathbf{A}$ to zero, can be constructed by simply inserting $-e_{i1} = -(a_{i1}/a_{11})$ at $(i, 1)'th$ location in an $n \times n$ identity matrix, i.e.

$$\mathbf{E}_{j1} = \begin{bmatrix} 1 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ -e_{i1} & ... & 1 & ... & 0 \\ 0 & ... & .... & ... & ... \\ 0 & ... & .... & 0 & 1 \end{bmatrix} \tag{21}$$

while the permutation matrix $\mathbf{P}_{ij}$, which interchanges i'th and j'th rows, can be created by interchanging i'th and j'th rows of the $n \times n$ identity matrix. The transformation matrix $\mathbf{T}$ for a general $n \times n$ matrix $\mathbf{A}$ can then be constructed by multiplying the elementary matrices, $\mathbf{E}_{ij}$, and the permutation matrices, $\mathbf{P}_{ij}$, in appropriate order such that $\mathbf{TA} = \mathbf{U}$.

It is important to note that the explanation presented in this subsection provides insights into the internal working of the Gaussian elimination process. While performing the Gaussian elimination on a particular matrix through a computer program, neither matrices $(\mathbf{E}_{ij}, \mathbf{P}_{ij})$ nor matrix $\mathbf{T}$ is constructed explicitly. For example, reducing elements $(i, 1)$ in the first column of matrix $\mathbf{A}$ to zero is achieved by performing the following set of computations

$$a_{ij}^{(i)} = a_{ij} - e_{i1}a_{1j} \quad \text{for} \quad j = 1, 2, ...n$$

where $i = 1, 2, ...n$. Performing these elimination calculations, which are carried out row wise and may require row exchanges, is *equivalent to constructing matrices* $(\mathbf{E}_{ij}, \mathbf{P}_{ij})$ *and effectively matrix* $\mathbf{T}$, *which is an invertible matrix.* Once we have reduced $\mathbf{Ax} = \mathbf{b}$ to $\mathbf{Ux} = \boldsymbol{\beta}$ form, such that $u_{ii} \neq 0$ for all i, then it is easy to recover the solution $\mathbf{x}$ using the back substitution method. Invertibility of matrix $\mathbf{T}$ guarantees that the solution of the transformed problem is identical to that of the original problem.

3.2 Number Computations in Direct Methods

Let $\boldsymbol{\varphi}$ denote the number of divisions and multiplications required for generating solution by a particular method. Various direct methods can be compared on the basis of $\boldsymbol{\varphi}$.

- **Cramers Rule:**

$$\boldsymbol{\varphi}(estimated) = (n-1)(n+1)(n!) + n \cong n^2 * n! \tag{22}$$

  For a problem of size $n = 100$ we have $\boldsymbol{\varphi} \cong 10^{162}$ and the time estimate for solving this problem on DEC1090 is approximately 10^{149}years [1].

- **Gaussian Elimination and Backward Sweep:** By maximal pivoting and row operations, we first reduce the system (5) to

$$\mathbf{Ux} = \widehat{\mathbf{b}} \tag{23}$$

  where $\mathbf{U}$ is a upper triangular matrix and then use backward sweep to solve (23) for $\mathbf{x}$.For this scheme, we have

$$\boldsymbol{\varphi} = \frac{n^3 + 3n^2 - n}{3} \cong \frac{n^3}{3} \tag{24}$$

  For $n = 100$ we have $\boldsymbol{\varphi} \cong 3.3 * 10^5$ [1].

- **LU-Decomposition:** LU decomposition is used when equation (5) is to be solved for several different values of vector $\mathbf{b}$, i.e.,

$$\mathbf{Ax}^{(k)} = \mathbf{b}^{(k)} \quad ; \quad k = 1, 2, ...N \tag{25}$$

  The sequence of computations is as follows

$$\mathbf{A} = \mathbf{LU} \quad \text{(Solved only once)} \tag{26}$$

$$\mathbf{Ly}^{(k)} = \mathbf{b}^{(k)} \quad ; \quad k = 1, 2, ...N \tag{27}$$

$$\mathbf{Ux}^{(k)} = \mathbf{y}^{(k)} \quad ; \quad k = 1, 2, ...N \tag{28}$$

For N different $\mathbf{b}$ vectors,

$$\boldsymbol{\varphi} = \frac{n^3 - n}{3} + Nn^2 \tag{29}$$

- **Gauss_Jordon Elimination:** In this case, we start with $[\mathbf{A} : I : b]$ and by sequence row operations we reduce it to $[I : \mathbf{A}^{-1} : x]$ i.e.

$$\left[\mathbf{A} : I : \mathbf{b}^{(k)}\right] \quad \underset{\text{Row Operations}}{\overset{\text{Sequence of}}{\rightarrow}} \quad \left[I : \mathbf{A}^{-1} : \mathbf{x}^{(k)}\right] \tag{30}$$

For this scheme, we have

$$\boldsymbol{\varphi} = \frac{n^3 + (N-1)n^2}{2} \tag{31}$$

Thus, Cramer's rule is certainly not suitable for numerical computations. The later three methods require significantly smaller number of multiplication and division operations when compared to Cramer's rule and are best suited for computing numerical solutions moderately large ($n \approx 1000$) systems. When number of equations is significantly large ($n \approx 10000$), even the Gaussian elimination and related methods can turn out to be computationally expensive and we have to look for alternative schemes that can solve (5) in smaller number of steps. When matrices have some **simple structure** (few non-zero elements and large number of zero elements), direct methods tailored to exploit sparsity and perform efficient numerical computations. Also, iterative methods give some hope as an approximate solution $\mathbf{x}$ can be calculated quickly using these techniques.

4 Direct Methods for Solving Sparse Linear Systems

A system of Linear equations

$$\mathbf{Ax} = \mathbf{b} \tag{32}$$

is called *sparse* if only a relatively small number of its matrix elements (a_{ij}) are nonzero. The sparse patterns that frequently occur are

- Tridiagonal
- Band diagonal with band width M
- Block diagonal matrices
- Lower / upper triangular and block lower / upper triangular matrices

We have encountered numerous situations in the module on Problem Discretization using Approximation Theory where such matrices arise. It is wasteful to apply general direct methods on these problems. Special methods are evolved for solving such sparse systems, which can achieve considerable reduction in the computation time and memory space requirements. In this section, some of the sparse matrix algorithms are discussed in detail. This is meant to be only a brief introduction to sparse matrix computations and the treatment of the topic is, by no means, exhaustive.

4.1 Block Diagonal Matrices

In some applications, such as solving ODE-BVP / PDE using orthogonal collocations on finite elements, we encounter equations with a block diagonal matrices, i.e.

$$\begin{bmatrix} \mathbf{A}_1 & [\mathbf{0}] & .... & [\mathbf{0}] \\ [\mathbf{0}] & \mathbf{A}_2 & .... & [\mathbf{0}] \\ ..... & . & .... & ..... \\ [\mathbf{0}] & [\mathbf{0}] & .... & \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ .... \\ \mathbf{x}^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ .... \\ \mathbf{b}^{(m)} \end{bmatrix}$$

where $\mathbf{A}_i$ for $i = 1, 2, ..m$ are $n_i \times n_i$ sub-matrices and $\mathbf{x}^{(i)} \in R^{n_i}$, $\mathbf{b}^{(i)} \in R^{n_i}$ for $i = 1, 2, ..m$ are sub-vectors. Such a system of equations can be solved by solving the following sub-problems

$$\mathbf{A}_i \mathbf{x}^{(i)} = \mathbf{b}^{(i)} \quad for \quad i = 1, 2, ...m \tag{33}$$

where each equation is solved using, say, Gaussian elimination. Each Gaussian elimination sub-problem would require

$$\boldsymbol{\varphi}_i = \frac{n_i^3 + 3n_i^2 - n_i}{3}$$

and

$$\boldsymbol{\varphi}(Block\ Diagonal) = \sum_{i=1}^{m} \boldsymbol{\varphi}_i$$

Defining dimension of vector $\mathbf{x}$ as $n = \sum_{i=1}^{m} n_i$, the number of multiplications and divisions in the conventional Gaussian elimination equals

$$\boldsymbol{\varphi}(Conventional) = \frac{n^3 + 3n^2 - n}{3}$$

It can be easily shown that

$$\sum_{i=1}^{m} \frac{n_i^3 + 3n_i^2 - n_i}{3} << \frac{\left(\sum_{i=1}^{m} n_i\right)^3 + 3\left(\sum_{i=1}^{m} n_i\right)^2 - \sum_{i=1}^{m} n_i}{3}$$

i.e.

$$\boldsymbol{\varphi}(Block\ Diagonal) << \boldsymbol{\varphi}(Conventional)$$

4.2 Thomas Algorithm for Tridiagonal and Block Tridiagonal Matrices [2]

Consider system of equation given by following equation

$$\begin{bmatrix} b_1 & c_1 & 0 & ... & ... & ... & ... & 0 \\ a_2 & b_2 & c_2 & 0 & ... & ... & ... & 0 \\ 0 & a_3 & b_3 & c_3 & ... & ... & ... & 0 \\ 0 & 0 & a_4 & b_4 & c_4. & ... & ... & ... \\ ... & ... & .. & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & c_{n-2} & 0 \\ ... & ... & ... & ... & ... & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & ... & .0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ .... \\ .... \\ .... \\ .... \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ .... \\ .... \\ .... \\ .... \\ .... \\ d_n \end{bmatrix} \tag{34}$$

where matrix **A** is a *tridiagonal* matrix. Thomas algorithm is the Gaussian elimination algorithm tailored to solve this type of sparse system.

- **Step 1:Triangularization:** Forward sweep with normalization

$$\gamma_1 = \frac{c_1}{b_1} \tag{35}$$

$$\gamma_k = \frac{c_k}{b_k - a_k\gamma_{k-1}} \quad for \quad k = 2, 3, ...(n-1) \tag{36}$$

$$\beta_1 = d_1/b_1 \tag{37}$$

$$\beta_k = \frac{(d_k - a_k\beta_{k-1})}{(b_k - a_k\gamma_{k-1})} \quad for \quad k = 2, 3, ...n \tag{38}$$

This sequence of operations finally results in the following system of equations

$$\begin{bmatrix} 1 & \gamma_1 & 0 & .... & 0 \\ 0 & 1 & \gamma_2 & .... & 0 \\ ... & 0 & 1 & .... & . \\ .... & .... & .... & .... & \gamma_{n-1} \\ 0 & 0 & ... & .... & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ .... \\ .... \\ x_n \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ \beta_n \end{bmatrix}$$

- **Step 2: Backward sweep** leads to solution vector

$$x_n = \beta_n$$

$$x_k = \beta_k - \gamma_k x_{k+1} \tag{39}$$

$$k = (n-1), (n-2), ..., 1 \tag{40}$$

Total no of multiplications and divisions in Thomas algorithm is

$$\varphi = 5n - 8$$

which is significantly smaller than the $n^3/3$ operations (approximately) necessary for carrying out the Gaussian elimination and backward sweep for a dense matrix.

The Thomas algorithm can be easily extended to solve a system of equations that involves a *block tridiagonal* matrix. Consider block tridiagonal system of the form

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 & [0]. & ..... & .... & [0]. \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & .... & ..... & ..... \\ .... & ..... & ..... & ..... & ..... & [0]. \\ ..... & ..... & ..... & ..... & .\mathbf{B}_{n-1} & \mathbf{C}_{n-1} \\ [0] & ..... & ..... & [0] & \mathbf{A}_n & \mathbf{B}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ . \\ . \\ \mathbf{x}^{(n)} \end{bmatrix} = \begin{bmatrix} \mathbf{d}^{(1)} \\ \mathbf{d}^{(2)} \\ . \\ . \\ \mathbf{d}^{(n)} \end{bmatrix} \tag{41}$$

where $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ are matrices and ($\mathbf{x}^{(i)}$, $\mathbf{d}^{(i)}$) represent vectors of appropriate dimensions.

- **Step 1:Block Triangularization**

$$\mathbf{\Gamma}_1 = [\mathbf{B}_1]^{-1} \mathbf{C}_1$$

$$\mathbf{\Gamma}_k = [\mathbf{B}_k - \mathbf{A}_k \Gamma_{k-1}]^{-1} \mathbf{C}_k \quad for \quad k = 2, 3, ...(n-1) \tag{42}$$

$$\boldsymbol{\beta}^{(1)} = [\mathbf{B}_1]^{-1} \mathbf{d}^{(1)} \tag{43}$$

$$\boldsymbol{\beta}^{(k)} = [\mathbf{B}_k - \mathbf{A}_k \mathbf{\Gamma}_{k-1}]^{-1} (\mathbf{d}^{(k)} - \mathbf{A}_k \boldsymbol{\beta}^{(k-1)}) \quad for \quad k = 2, 3, ...n$$

- **Step 2: Backward sweep**

$$\mathbf{x}^{(n)} = \boldsymbol{\beta}^{(n)} \tag{44}$$

$$\mathbf{x}^{(k)} = \boldsymbol{\beta}^{(k)} - \mathbf{\Gamma}_k \mathbf{x}^{(k+1)} \tag{45}$$

$$\text{for } k = (n-1), (n-2), ..., 1 \tag{46}$$

4.3 Triangular and Block Triangular Matrices

A triangular matrix is a sparse matrix with zero-valued elements above or below the diagonal. For example, a lower triangular matrix can be represented as follows

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & . & . & 0 \\ l_{12} & l_{22} & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ l_{\ n1} & . & . & . & l_{\ nn} \end{bmatrix}$$

To solve a system $\mathbf{Lx} = \mathbf{b}$, the following algorithm is used

$$x_1 = \frac{b_1}{l_{11}} \tag{47}$$

$$x_i = \frac{1}{l_{ii}} \left[b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right] \quad for \quad i = 2, 3, ...n \tag{48}$$

The operational count $\boldsymbol{\varphi}$ i.e., the number of multiplications and divisions, for this elimination process is

$$\boldsymbol{\varphi} = \frac{n(n+1)}{2} \tag{49}$$

which is considerably smaller than the Gaussian elimination for a dense matrix.

In some applications we encounter equations with a block triangular matrices. For example,

$$\begin{bmatrix} \mathbf{A}_{1,1} & [0] & .... & [0] \\ \mathbf{A}_{1,2} & \mathbf{A}_{2,2} & .... & [0] \\ ..... & . & .... & ..... \\ \mathbf{A}_{m,1} & \mathbf{A}_{m,2} & .... & \mathbf{A}_{m,m} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ .... \\ \mathbf{x}^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ .... \\ \mathbf{b}^{(m)} \end{bmatrix}$$

where $\mathbf{A}_{i,j}$ are $n_i \times n_i$ sub-matrices while $\mathbf{x}^{(i)} \in R^{n_i}$ and $\mathbf{b}^{(i)} \in R^{n_i}$ are sub-vectors for $i = 1, 2, ..m$. The solution of this type of systems is completely analogous to that of lower triangular systems,except that sub-matrices and sub-vectors are used in place of scalars. Thus, the equivalent algorithm for block-triangular matrix can be stated as follows

$$\boldsymbol{\eta}^{(1)} = (\mathbf{A}_{11})^{-1} \mathbf{b}^{(1)} \tag{50}$$

$$\mathbf{x}^{(i)} = (\mathbf{A}_{ii})^{-1} [\mathbf{b}^{(i)} - \sum_{j=1}^{i-1} (\mathbf{A}_{ij} \mathbf{x}^{(i)})] \quad for \quad i = 2, 3, ...m \tag{51}$$

The above form does not imply that the inverse $(\mathbf{A}_{ii})^{-1}$should be computed explicitly. For example, we can find each $\mathbf{x}^{(i)}$ by Gaussian elimination.

4.4 Solution of a Large System By Partitioning

If matrix $\mathbf{A}$ is equation (5) is very large, then we can partition matrix $\mathbf{A}$ and vector $\mathbf{b}$ as

$$\mathbf{Ax} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \end{bmatrix}$$

where $\mathbf{A}_{11}$ is a $(m \times m)$ square matrix. this results in two equations

$$\mathbf{A}_{11}\mathbf{x}^{(1)} + \mathbf{A}_{12}\mathbf{x}^{(2)} = \mathbf{b}^{(1)} \tag{52}$$

$$\mathbf{A}_{21}\mathbf{x}^{(1)} + \mathbf{A}_{22}\,\mathbf{x}^{(2)} = \mathbf{b}^{(2)} \tag{53}$$

which can be solved sequentially as follows

$$\mathbf{x}^{(1)} = [\mathbf{A}_{11}]^{-1}\left[\mathbf{b}^{(1)} - \mathbf{A}_{12}\mathbf{x}^{(2)}\right] \tag{54}$$

$$\mathbf{A}_{21}\left[\mathbf{A}_{11}\right]^{-1}\left[\mathbf{b}^{(1)} - \mathbf{A}_{12}\mathbf{x}^{(2)}\right] + \mathbf{A}_{22}\mathbf{x}^{(2)} = \mathbf{b}^{(2)} \tag{55}$$

$$\left[\mathbf{A}_{22} - \mathbf{A}_{21}\left[\mathbf{A}_{11}\right]^{-1}\mathbf{A}_{12}\right]\mathbf{x}^{(2)} = \mathbf{b}^{(2)} - (\mathbf{A}_{21}\mathbf{A}_{11}^{-1})\mathbf{b}^{(1)} \tag{56}$$

$$\mathbf{x}^{(2)} = \left[\mathbf{A}_{22} - \mathbf{A}_{21}\left[\mathbf{A}_{11}\right]^{-1}\mathbf{A}_{12}\right]^{-1}\left[\mathbf{b}^{(2)} - (\mathbf{A}_{21}\mathbf{A}_{11}^{-1})\mathbf{b}^{(1)}\right]$$

It is also possible to work with higher number of partitions equal to say 9, 16 .... and solve a given large dimensional problem.

5 Iterative Solution Techniques

By this approach, we start with some initial *guess solution*, say $\mathbf{x}^{(0)}$, for solution $\mathbf{x}$ and generate an improved solution estimate $\mathbf{x}^{(k+1)}$ from the previous approximation $\mathbf{x}^{(k)}$. This method is a very effective for solving differential equations, integral equations and related problems [4]. Let the residue vector $\mathbf{r}$ be defined as

$$r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij}x_j^{(k)} \text{ for } i = 1, 2, ...n \tag{57}$$

i.e. $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$. The iteration sequence $\left\{\mathbf{x}^{(k)} : k = 0, 1, ...\right\}$ is terminated when some norm of the residue $\left\|\mathbf{r}^{(k)}\right\| = \left\|\mathbf{A}\mathbf{x}^{(k)} - \mathbf{b}\right\|$ becomes sufficiently small, i.e.

$$\frac{\left\|\mathbf{r}^{(k)}\right\|}{\|\mathbf{b}\|} < \varepsilon \tag{58}$$

where ε is an arbitrarily small number (such as 10^{-8} or 10^{-10}). Another possible termination criterion can be

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\right\|}{\left\|\mathbf{x}^{(k+1)}\right\|} < \varepsilon \tag{59}$$

It may be noted that the later condition is *practically* equivalent to the previous termination condition.

A simple way to form an iterative scheme is *Richardson iterations* [4]

$$\mathbf{x}^{(k+1)} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{(k)} + \mathbf{b} \tag{60}$$

or *Richardson iterations* preconditioned with approximate inversion

$$\mathbf{x}^{(k+1)} = (\mathbf{I} - \mathbf{MA})\mathbf{x}^{(k)} + \mathbf{Mb} \tag{61}$$

where matrix $\mathbf{M}$ is called approximate inverse of $\mathbf{A}$ if $\|\mathbf{I} - \mathbf{MA}\| < 1$. A question that naturally arises is '*will the iterations converge to the solution of* $\mathbf{Ax} = \mathbf{b}$*?*'. In this section, to begin with, some well known iterative schemes are presented. Their convergence analysis is presented next. In the derivations that follow, it is implicitly assumed that the diagonal elements of matrix $\mathbf{A}$ are non-zero, i.e. $a_{ii} \neq 0$. If this is not the case, simple row exchange is often sufficient to satisfy this condition.

5.1 Iterative Algorithms

5.1.1 Jacobi-Method

Suppose we have a guess solution, say $\mathbf{x}^{(k)}$,

$$\mathbf{x}^{(k)} = \begin{bmatrix} x_1^{(k)} & x_2^{(k)} & .... & x_n^{(k)} \end{bmatrix}^T$$

for $\mathbf{Ax} = \mathbf{b}$. To generate an improved estimate starting from $\mathbf{x}^{(k)}$,consider the first equation in the set of equations $\mathbf{Ax} = \mathbf{b}$, i.e.,

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \tag{62}$$

Rearranging this equation, we can arrive at a iterative formula for computing $x_1^{(k+1)}$, as

$$x_1^{(k+1)} = \frac{1}{a_{11}}\left[b_1 - a_{12}x_2^{(k)} ..... - a_{1n}x_n^{(k)}\right] \tag{63}$$

Similarly, using second equation from $\mathbf{Ax} = \mathbf{b}$, we can derive

$$x_2^{(k+1)} = \frac{1}{a_{22}}\left[b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} ..... - a_{2n}x_n^{(k)}\right] \tag{64}$$

Table 1: Algorithm for Jacobi Iterations

INITIALIZE : $\mathbf{b}, \mathbf{A}, \mathbf{x}^{(0)}, k_{\max}, \varepsilon$

$k = 0$

$\delta = 100 * \varepsilon$

WHILE $[(\delta > \varepsilon) \;\; AND \;\; (k < k_{\max})]$

FOR $i = 1 : n$

$$r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$$

$$x_{Ni} = x_i + \left(\frac{r_i}{a_{ii}} \right)$$

END FOR

$$\delta = \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

$\times = \mathbf{x}_N$

$k = k + 1$

END WHILE

In general, using i^{th}row of $\mathbf{Ax} = \mathbf{b}$, we can generate improved guess for the i'th element $\mathbf{x}$ of as follows

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_2 - a_{i1} x_1^{(k)} ..... - a_{i,i-1} x_{i-1}^{(k)} - a_{i,i+1} x_{i+1}^{(k)} ..... - a_{i,n} x_n^{(k)} \right] \tag{65}$$

The above equation can also be rearranged as follows

$$x_i^{(k+1)} = x_i^{(k)} + \left(\frac{r_i^{(k)}}{a_{ii}} \right)$$

where $r_i^{(k)}$ is defined by equation (57). The algorithm for implementing the Jacobi iteration scheme is summarized in Table 1.

5.1.2 Gauss-Seidel Method

When matrix $\mathbf{A}$ is large, there is a practical difficulty with the Jacobi method. It is required to store all components of $\mathbf{x}^{(k)}$ in the computer memory (as a separate variables) until calculations of $\mathbf{x}^{(k+1)}$ is over. The Gauss-Seidel method overcomes this difficulty by using $x_i^{(k+1)}$ immediately in the next equation while computing $x_{i+1}^{(k+1)}$.This modification leads to the following set of equations

$$x_1^{(k+1)} = \frac{1}{a_{11}} \left[b_1 - a_{12} x_2^{(k)} - a_{13} x_3^{(k)} .....a_{1n} x_n^{(k)} \right] \tag{66}$$

Table 2: Algorithm for Gauss-Seidel Iterations

INITIALIZE :$\mathbf{b}, \mathbf{A}, \mathbf{x}, k_{\max}, \varepsilon$
$k = 0$
$\delta = 100 * \varepsilon$
WHILE $[(\delta > \varepsilon) \;\; AND \;\; (k < k_{\max})]$
    FOR $i = 1 : n$
        $r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$
        $x_i = x_i + (r_i / a_{ii})$
    END FOR
    $\delta = \|\mathbf{r}\| \,/\, \|\mathbf{b}\|$
    $k = k + 1$
END WHILE

$$x_2^{(k+1)} = \frac{1}{a_{22}} \left[b_2 - \left\{ a_{21} x_1^{(k+1)} \right\} - \left\{ a_{23} x_3^{(k)} + ... + a_{2n} x_n^{(k)} \right\} \right] \tag{67}$$

$$x_3^{(k+1)} = \frac{1}{a_{33}} \left[b_3 - \left\{ a_{31} x_1^{(k+1)} + a_{32} x_2^{(k+1)} \right\} - \left\{ a_{34} x_4^{(k)} + ... + a_{3n} x_n^{(k)} \right\} \right] \tag{68}$$

In general, for i'th element of $\mathbf{x}$, we have

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

To simplify programming, the above equation can be rearranged as follows

$$x_i^{(k+1)} = x_i^{(k)} + \left(r_i^{(k)} / a_{ii} \right) \tag{69}$$

where

$$r_i^{(k)} = \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)} \right]$$

The algorithm for implementing Gauss-Siedel iteration scheme is summarized in Table 2.

5.1.3 Relaxation Method

Suppose we have a starting value say y, of a quantity and we wish to approach a target value, say y^*, by some method. Let application of the method change the value from y to $\widehat{y}$. If $\widehat{y}$ is between y and $\widetilde{y}$, which is even closer to y^* than $\widehat{y}$, then we can approach y^* faster by

Table 3: Algorithms for Over-Relaxation Iterations

INITIALIZE : $\mathbf{b}, \mathbf{A}, \mathbf{x}, k_{\max}, \varepsilon, \omega$
  $k = 0$
  $\delta = 100 * \varepsilon$
WHILE $[(\delta > \varepsilon) \;\; AND \;\; (k < k_{\max})]$
  FOR $i = 1 : n$
    $r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$
    $z_i = x_i + (r_i / a_{ii})$
    $x_i = x_i + \omega(z_i - x_i)$
  END FOR
  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
  $\delta = \|\mathbf{r}\| \, / \, \|\mathbf{b}\|$
  $k = k + 1$
END WHILE

magnifying the change $(\widehat{y} - y)$ [3]. In order to achieve this, we need to apply a magnifying factor $\omega > 1$ and get

$$\widetilde{y} = y + \omega \, (\widehat{y} - y) \tag{70}$$

This amplification process is an *extrapolation* and is an example of **over-relaxation**. If the intermediate value $\widehat{y}$ tends to overshoot target y^*, then we may have to use $\omega < 1$; this is called **under-relaxation**.

Application of **over-relaxation** to Gauss-Seidel method leads to the following set of equations

$$\begin{aligned} x_i^{(k+1)} &= x_i^{(k)} + \omega[z_i^{(k+1)} - x_i^{(k)}] \\ i &= 1, 2, ...n \end{aligned} \tag{71}$$

where $z_i^{(k+1)}$ are generated using the Gauss-Seidel method, i.e.,

$$\begin{aligned} z_i^{(k+1)} &= \left(\frac{1}{a_{ii}}\right) \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] \\ i &= 1, 2, ...n \end{aligned} \tag{72}$$

The steps in the implementation of the over-relaxation iteration scheme are summarized in Table 3. It may be noted that ω is a *tuning parameter,* which is chosen such that $1 < \omega < 2$.

5.2 Convergence Analysis of Iterative Methods [3, 2]

5.2.1 Vector-Matrix Representation

When $\mathbf{Ax} = \mathbf{b}$ is to be solved iteratively, a question that naturally arises is 'under what conditions the iterations converge?'. The convergence analysis can be carried out if the above set of iterative equations are expressed in the vector-matrix notation. For example, the iterative equations in the Gauss-Siedel method can be arranged as follows

$$\begin{bmatrix} a_{11} & 0 & .. & 0 \\ a_{21} & a_{22} & 0 & . \\ ... & ... & ... & 0 \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ ... \\ ... \\ x_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} 0 & -a_{12} & -a_{13} & ... & -a_{1,n} \\ 0 & 0 & -a_{23} & ... & -a_{2,n} \\ . & . & . & .. & -a_{n-1,n} \\ 0 & . & . & . & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ . \\ . \\ x_n^{(k)} \end{bmatrix} + \begin{bmatrix} b_1 \\ . \\ . \\ b_n \end{bmatrix} \tag{73}$$

Let $\mathbf{D}, \mathbf{L}$ and $\mathbf{U}$ be diagonal, strictly lower triangular and strictly upper triangular parts of $\mathbf{A}$, i.e.,

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U} \tag{74}$$

(*The representation given by equation 74 should NOT be confused with matrix factorization* $\mathbf{A} = \mathbf{LDU}$). Using these matrices, the Gauss-Siedel iteration can be expressed as follows

$$(\mathbf{L} + \mathbf{D})\mathbf{x}^{(k+1)} = -\mathbf{U}\mathbf{x}^{(k)} + \mathbf{b} \tag{75}$$

or

$$\mathbf{x}^{(k+1)} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{L} + \mathbf{D})^{-1}\mathbf{b} \tag{76}$$

Similarly, rearranging the iterative equations for Jacobi method, we arrive at

$$\mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b} \tag{77}$$

and for the relaxation method we get

$$\mathbf{x}^{(k+1)} = (\mathbf{D} + \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{D} - \omega\mathbf{U}]\mathbf{x}^{(k)} + \omega\,\mathbf{b} \tag{78}$$

Thus, in general, an iterative method can be developed by splitting matrix $\mathbf{A}$. If $\mathbf{A}$ is expressed as

$$\mathbf{A} = \mathbf{S} - \mathbf{T} \tag{79}$$

then, equation $\mathbf{Ax} = \mathbf{b}$ can be expressed as

$$\mathbf{Sx} = \mathbf{Tx} + \mathbf{b}$$

Starting from a guess solution

$$\mathbf{x}^{(0)} = [x_1^{(0)}.....x_n^{(0)}]^T \tag{80}$$

we generate a sequence of *approximate solutions* as follows

$$\mathbf{x}^{(k+1)} = \mathbf{S}^{-1}[\mathbf{T}\mathbf{x}^{(k)} + \mathbf{b}] \quad where \quad k = 0, 1, 2, ... \tag{81}$$

Requirements on $\mathbf{S}$ and $\mathbf{T}$ matrices are as follows [3] : matrix A should be decomposed into $\mathbf{A} = \mathbf{S} - \mathbf{T}$ such that

- Matrix $\mathbf{S}$ should be easily invertible
- Sequence $\left\{\mathbf{x}^{(k)} : k = 0, 1, 2, ...\right\}$ should converge to $\mathbf{x}^*$ where $\mathbf{x}^*$ is the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$.

The popular iterative formulations correspond to the following choices of matrices $\mathbf{S}$ and $\mathbf{T}$[3, 4]

- **Jacobi Method:**

$$\mathbf{S}_{JAC} = \mathbf{D}, \quad \mathbf{T}_{JAC} = -(\mathbf{L} + \mathbf{U}) \tag{82}$$

- **Forward Gauss-Seidel Method**

$$\mathbf{S}_{GS} = \mathbf{L} + \mathbf{D}, \quad \mathbf{T}_{GS} = -\mathbf{U} \tag{83}$$

- **Relaxation Method:**

$$\mathbf{S}_{SOR} = \omega\mathbf{L} + \mathbf{D}, \quad \mathbf{T}_{SOR} = (1 - \omega)\,\mathbf{D} - \omega\mathbf{U} \tag{84}$$

- **Backward Gauss Seidel:** In this case, iteration begins the update of $\mathbf{x}$ with n'th coordinate rather than the first. This results in the following splitting of matrix $\mathbf{A}$ [4]

$$\mathbf{S}_{BSG} = \mathbf{U} + \mathbf{D}, \quad \mathbf{T}_{BSG} = -\mathbf{L} \tag{85}$$

In **Symmetric Gauss Seidel** approach**,** a forward Gauss-Seidel iteration is followed by a backward Gauss-Seidel iteration.

5.2.2 Iterative Scheme as a Linear Difference Equation

In order to solve equation (5), we have formulated an iterative scheme

$$\mathbf{x}^{(k+1)} = \left(\mathbf{S}^{-1}\mathbf{T}\right)\,\mathbf{x}^{(k)} + \mathbf{S}^{-1}\mathbf{b} \tag{86}$$

Let the true solution equation (5) be

$$\mathbf{x}^* = \left(\mathbf{S}^{-1}\mathbf{T}\right)\mathbf{x}^* + \mathbf{S}^{-1}\mathbf{b} \tag{87}$$

Defining error vector

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^{*} \tag{88}$$

and subtracting equation (87) from equation (86), we get

$$\mathbf{e}^{(k+1)} = \left(\mathbf{S}^{-1}\mathbf{T}\right)\, \mathbf{e}^{(k)} \tag{89}$$

Thus, if we start with some $\mathbf{e}^{(0)}$, then after k iterations we have

$$\begin{aligned}
\mathbf{e}^{(1)} &= \left(\mathbf{S}^{-1}\mathbf{T}\right) \mathbf{e}^{(0)} && (90)\\
\mathbf{e}^{(2)} &= \left(\mathbf{S}^{-2}\mathbf{T}^{2}\right) \mathbf{e}^{(1)} = [\mathbf{S}^{-1}\mathbf{T}]^{2}\mathbf{e}^{(0)} && (91)\\
.... &= ..... && (92)\\
\mathbf{e}^{(k)} &= [\mathbf{S}^{-1}\mathbf{T}]^{k}\mathbf{e}^{(0)} && (93)
\end{aligned}$$

The convergence of the iterative scheme is assured if

$$\lim_{k \to \infty} \mathbf{e}^{(k)} = \overline{\mathbf{0}} \tag{94}$$

$$\text{i.e.} \quad \lim_{k \to \infty} [\mathbf{S}^{-1}\mathbf{T}]^{k}\mathbf{e}^{(0)} = \overline{\mathbf{0}} \tag{95}$$

for **any** initial guess vector $\mathbf{e}^{(0)}$.

Alternatively, consider application of the general iteration equation (86) k times starting from initial guess $\mathbf{x}^{(0)}$. At the k'th iteration step, we have

$$\mathbf{x}^{(k)} = \left(\mathbf{S}^{-1}\mathbf{T}\right)^{k} \mathbf{x}^{(0)} + \left[\left(\mathbf{S}^{-1}\mathbf{T}\right)^{k-1} + \left(\mathbf{S}^{-1}\mathbf{T}\right)^{k-2} + ... + \mathbf{S}^{-1}\mathbf{T} + \mathbf{I}\right] \mathbf{S}^{-1}\mathbf{b} \tag{96}$$

If we select $(\mathbf{S}^{-1}\mathbf{T})$ such that

$$\lim_{k \to \infty} \left(\mathbf{S}^{-1}\mathbf{T}\right)^{k} \to [\mathbf{0}] \tag{97}$$

where $[\mathbf{0}]$ represents the null matrix, then, using identity

$$\left[\mathbf{I} - \left(\mathbf{S}^{-1}\mathbf{T}\right)\right]^{-1} = \mathbf{I} + \left(\mathbf{S}^{-1}\mathbf{T}\right) + ... + \left(\mathbf{S}^{-1}\mathbf{T}\right)^{k-1} + \left(\mathbf{S}^{-1}\mathbf{T}\right)^{k} + ...$$

we can write

$$\mathbf{x}^{(k)} \to \left[\mathbf{I} - \left(\mathbf{S}^{-1}\mathbf{T}\right)\right]^{-1} \mathbf{S}^{-1}\mathbf{b} = [\mathbf{S} - \mathbf{T}]^{-1}\, \mathbf{b} = \mathbf{A}^{-1}\mathbf{b}$$

for large k. The above expression clearly explains how the iteration sequence generates a numerical approximation to $\mathbf{A}^{-1}\mathbf{b}$, provided condition (97) is satisfied.

5.2.3 Convergence Criteria for Iteration Schemes

It may be noted that equation (89) is a linear difference equation of form

$$\mathbf{z}^{(k+1)} = \mathbf{B}\mathbf{z}^{(k)} \tag{98}$$

with a specified initial condition $\mathbf{z}^{(0)}$. Here, $\mathbf{z} \in \mathbf{R}^n$ and $\mathbf{B}$ is a $n \times n$ matrix. In Appendix A, we analyzed behavior of the solutions of linear difference equations of type (98). The criterion for convergence of iteration equation (89) can be derived using results derived in Appendix A. The necessary and sufficient condition for convergence of (89) can be stated as

$$\rho(\mathbf{S}^{-1}\mathbf{T}) < 1$$

i.e. the spectral radius of matrix $\mathbf{S}^{-1}\mathbf{T}$ should be less than one.

The necessary and sufficient condition for convergence stated above requires computation of eigenvalues of $\mathbf{S}^{-1}\mathbf{T}$, which is a computationally demanding task when the matrix dimension is large. For a large dimensional matrix, if we could check this condition before starting the iterations, then we might as well solve the problem by a direct method rather than using iterative approach to save computations. Thus, there is a need to derive some alternate criteria for convergence, which can be checked easily before starting iterations. Theorem 14 in Appendix A states that spectral radium of a matrix is smaller than any induced norm of the matrix. Thus, for matrix $\mathbf{S}^{-1}\mathbf{T}$, we have

$$\rho\left(\mathbf{S}^{-1}\mathbf{T}\right) \leq \left\|\mathbf{S}^{-1}\mathbf{T}\right\|$$

$\|.\|$ is any induced matrix norm. Using this result, we can arrive at the following sufficient conditions for the convergence of iterations

$$\left\|\mathbf{S}^{-1}\mathbf{T}\right\|_1 < 1 \quad \text{or} \quad \left\|\mathbf{S}^{-1}\mathbf{T}\right\|_\infty < 1$$

Evaluating 1 or ∞ norms of $\mathbf{S}^{-1}\mathbf{T}$ is significantly easier than evaluating the spectral radius of $\mathbf{S}^{-1}\mathbf{T}$. Satisfaction of any of the above conditions implies $\rho\left(\mathbf{S}^{-1}\mathbf{T}\right) < 1$. However, it may be noted that these are only sufficient conditions. Thus, if $\left\|\mathbf{S}^{-1}\mathbf{T}\right\|_\infty > 1$ or $\left\|\mathbf{S}^{-1}\mathbf{T}\right\|_1 > 1$, we cannot conclude anything about the convergence of iterations.

If the matrix $\mathbf{A}$ has some special properties, such as diagonal dominance or symmetry and positive definiteness, then the convergence is ensured for some iterative techniques. Some of the important convergence results available in the literature are summarized here.

Definition 1 *A matrix* $\mathbf{A}$ *is called strictly diagonally dominant if*

$$\sum_{j=1(j\neq i)}^{n} |a_{ij}| < |a_{ii}| \quad \textit{for} \;\; i = 1, 2., ...n \tag{99}$$

Theorem 2 *[2] A sufficient condition for the convergence of Jacobi and Gauss-Seidel methods is that the matrix* $\mathbf{A}$ *of linear system* $\mathbf{Ax} = \mathbf{b}$ *is strictly diagonally dominant.*

Proof: *Refer to Appendix B.*

Theorem 3 *[5] The Gauss-Seidel iterations converge if matrix* $\mathbf{A}$ *is symmetric and positive definite.*

Proof: *Refer to Appendix B.*

Theorem 4 *[3] For an arbitrary matrix* $\mathbf{A}$*, the necessary condition for the convergence of relaxation method is* $0 < \omega < 2$.

Proof: *Refer to appendix B.*

Theorem 5 *[2] When matrix* $\mathbf{A}$ *is strictly diagonally dominant, a sufficient condition for the convergence of relaxation methods is that* $0 < \omega \leq 1$.

Proof: *Left to reader as an exercise.*

Theorem 6 *[2]For a symmetric and positive definite matrix* $\mathbf{A}$*, the relaxation method converges if and only if* $0 < \omega < 2$.

Proof: *Left to reader as an exercise.*

The Theorems 3 and 6 guarantees convergence of Gauss-Seidel method or relaxation method when matrix $\mathbf{A}$ is symmetric and positive definite. Now, what do we do if matrix $\mathbf{A}$ in $\mathbf{Ax} = \mathbf{b}$ is not symmetric and positive definite? We can multiply both the sides of the equation by $\mathbf{A}^T$ and transform the original problem as follows

$$\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} = \left(\mathbf{A}^T\mathbf{b}\right) \tag{100}$$

If matrix $\mathbf{A}$ is non-singular, then matrix $\left(\mathbf{A}^T\mathbf{A}\right)$ is always symmetric and positive definite as

$$\mathbf{x}^T\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} = \left(\mathbf{Ax}\right)^T\left(\mathbf{Ax}\right) > 0 \text{ for any } \mathbf{x} \neq \overline{0} \tag{101}$$

Now, for the transformed problem, we are guaranteed convergence if we use the Gauss-Seidel method or relaxation method such that $0 < \omega < 2$.

Example 7 *[3]Consider system* $\mathbf{Ax} = \mathbf{b}$ *where*

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \tag{102}$$

For Jacobi method

$$\mathbf{S}^{-1}\mathbf{T} = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix} \tag{103}$$

$$\rho(\mathbf{S}^{-1}\mathbf{T}) = 1/2 \tag{104}$$

Thus, the error norm at each iteration is reduced by factor of 0.5For Gauss-Seidel method

$$\mathbf{S}^{-1}\mathbf{T} = \begin{bmatrix} 0 & 1/2 \\ 0 & 1/4 \end{bmatrix} \tag{105}$$

$$\rho(\mathbf{S}^{-1}\mathbf{T}) = 1/4 \tag{106}$$

Thus, the error norm at each iteration is reduced by factor of 1/4. This implies that, for the example under consideration

$$\textit{1 Gauss Seidel iteration} \equiv \textit{2 Jacobi iterations} \tag{107}$$

For relaxation method,

$$\mathbf{S}^{-1}\mathbf{T} = \begin{bmatrix} 2 & 0 \\ -\omega & 2 \end{bmatrix}^{-1} \begin{bmatrix} 2(1-\omega) & \omega \\ 0 & 2(1-\omega) \end{bmatrix} \tag{108}$$

$$= \begin{bmatrix} (1-\omega) & (\omega/2) \\ (\omega/2)(1-\omega) & (1-\omega+\frac{\omega^2}{4}) \end{bmatrix} \tag{109}$$

$$\lambda_1\lambda_2 = \det(\mathbf{S}^{-1}\mathbf{T}) = (1-\omega)^2 \tag{110}$$

$$\lambda_1 + \lambda_2 = trace(\mathbf{S}^{-1}\mathbf{T}) \tag{111}$$

$$= 2 - 2\omega + \frac{\omega^2}{4} \tag{112}$$

Now, if we plot $\rho(\mathbf{S}^{-1}\mathbf{T})$ v/s ω, then it is observed that $\lambda_1 = \lambda_2$ at $\omega = \omega_{opt}$.From equation (110), it follows that

$$\lambda_1 = \lambda_2 = \omega_{opt} - 1 \tag{113}$$

at optimum ω.Now,

$$\lambda_1 + \lambda_2 = 2(\omega_{opt} - 1) \tag{114}$$

$$= 2 - 2\omega_{opt} + \frac{\omega_{opt}^2}{4} \tag{115}$$

$$\Rightarrow \quad \omega_{opt} = 4(2-\sqrt{3}) \cong 1.07 \tag{116}$$

$$\Rightarrow \quad \rho(\mathbf{S}^{-1}\mathbf{T}) = \lambda_1 = \lambda_2 \cong 0.07 \tag{117}$$

This is a major reduction in spectral radius when compared to Gauss-Seidel method. Thus, the error norm at each iteration is reduced by factor of 1/16 ($\cong 0.07$) if we choose $\omega = \omega_{opt}$.

Example 8 *Consider system* $\mathbf{Ax} = \mathbf{b}$ *where*

$$\mathbf{A} = \begin{bmatrix} 4 & 5 & 9 \\ 7 & 1 & 6 \\ 5 & 2 & 9 \end{bmatrix} \quad ; \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{118}$$

If we use Gauss-Seidel method to solve for $\mathbf{x}$*, the iterations do not converge as*

$$\mathbf{S}^{-1}\mathbf{T} = \begin{bmatrix} 4 & 0 & 0 \\ 7 & 1 & 0 \\ 5 & 2 & 9 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -5 & -9 \\ 0 & 0 & -6 \\ 0 & 0 & 0 \end{bmatrix} \tag{119}$$

$$\rho(\mathbf{S}^{-1}\mathbf{T}) = 7.3 > 1 \tag{120}$$

Now, let us modify the problem by pre-multiplying $\mathbf{Ax} = \mathbf{b}$ *by* $\mathbf{A}^T$ *on both the sides, i.e. the modified problem is* $\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} = \left(\mathbf{A}^T\mathbf{b}\right)$*. The modified problem becomes*

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 90 & 37 & 123 \\ 37 & 30 & 69 \\ 123 & 69 & 198 \end{bmatrix} ; \quad \mathbf{A}^T\mathbf{b} = \begin{bmatrix} 16 \\ 8 \\ 24 \end{bmatrix} \tag{121}$$

The matrix $\mathbf{A}^T\mathbf{A}$ *is symmetric and positive definite and, according to Theorem 3, the iterations should converge if Gauss-Seidel method is used. For the transformed problem, we have*

$$\mathbf{S}^{-1}\mathbf{T} = \begin{bmatrix} 90 & 0 & 0 \\ 37 & 30 & 0 \\ 123 & 69 & 198 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -37 & -123 \\ 0 & 0 & -69 \\ 0 & 0 & 0 \end{bmatrix} \tag{122}$$

$$\rho(\mathbf{S}^{-1}\mathbf{T}) = 0.96 < 1 \tag{123}$$

and within 220 iterations (termination criterion 1×10^{-5}*), we get following solution*

$$\mathbf{x} = \begin{bmatrix} 0.0937 \\ 0.0312 \\ 0.0521 \end{bmatrix} \tag{124}$$

which is close to the solution

$$\mathbf{x}^* = \begin{bmatrix} 0.0937 \\ 0.0313 \\ 0.0521 \end{bmatrix} \tag{125}$$

computed as $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.

Table 4: Rate of Convergence of Iterative Methods

Method	Rate of Convergence	No. of iterations for ε error
Jacobi	$O(1/2n^2)$	$O(2n^2)$
Gauss_Seidel	$O(1/n^2)$	$O(n^2)$
Relaxation with optimal ω	$O(2/n)$	$O(n/2)$

Example 9 *Consider system* $\mathbf{Ax} = \mathbf{b}$ *where*

$$\mathbf{A} = \begin{bmatrix} 7 & 1 & -2 & 1 \\ 1 & 8 & 1 & 0 \\ -2 & 1 & 5 & -1 \\ 1 & 0 & -1 & 3 \end{bmatrix} \quad ; \quad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \tag{126}$$

If it is desired to solve the resulting problem using Jacobi method / Gauss-Seidel method, will the iterations converge? To establish convergence of Jacobi / Gauss-Seidel method, we can check whether A is strictly diagonally dominant. Since the following inequalities hold

$$\begin{aligned} \textit{Row}\ \ 1: &\ \ 1 + |-2| + 1 < 7 \\ \textit{Row}\ \ 2: &\ \ 1 + 0 + 1 < 8 \\ \textit{Row}\ \ 3: &\ \ |-2| + 1 + |-1| < 5 \\ \textit{Row}\ \ 4: &\ \ 1 + 0 + |-1| < 3 \end{aligned}$$

matrix A is strictly diagonally dominant, which is a sufficient condition for convergence of Jacobi / Gauss-Seidel iterations Theorem 2. Thus, Jacobi / Gauss-Seidel iterations will converge to the solution starting from ***any*** *initial guess.*

From these examples, we can clearly see that the rate of convergence depends on $\rho(\mathbf{S}^{-1}\mathbf{T})$. A comparison of rates of convergence obtained from analysis of some simple problems is presented in Table 4[2].

6 Optimization Based Methods

Unconstrained optimization is another tool employed to solve large scale linear algebraic equations. Gradient and conjugate gradient methods for numerically solving unconstrained optimization problems can be tailored to solve a set of linear algebraic equations. The development of the gradient search method for unconstrained optimization for any scalar objective function $\phi(\mathbf{x}) : R^n \rightarrow R$ is presented in Appendix C. In this section, we present how this method can be tailored for solving $\mathbf{Ax} = \mathbf{b}$.

6.1 Gradient Search Method

Consider system of linear algebraic equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \;\; ; \;\; \mathbf{x}, \mathbf{b} \in \mathbf{R}^n \tag{127}$$

where $\mathbf{A}$ is a non-singular matrix. Defining objective function

$$\phi(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{128}$$

the necessary condition for optimality requires that

$$\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \overline{\mathbf{0}} \tag{129}$$

Since $\mathbf{A}$ is assumed to be nonsingular, the stationarity condition is satisfied only at the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$. The stationary point is also a minimum as $\left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^2}\right] = \mathbf{A}^T\mathbf{A}$ is a positive definite matrix. Thus, we can compute the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ by minimizing

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} - \left(\mathbf{A}^T\mathbf{b}\right)^T \mathbf{x} \tag{130}$$

When $\mathbf{A}$ is positive definite, the minimization problem can be formulated as follows

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{131}$$

as it can be shown that the above function achieves a minimum for x* where Ax*=b. In the development that follows, for the sake of simplifying the notation, it is assumed that $\mathbf{A}$ is symmetric and positive definite. When the original problem does not involve a symmetric positive definite matrix, then it can always be transformed by pre-multiplying both sides of the equation by $\mathbf{A}^T$.

To arrive at the gradient search algorithm, given a guess solution $\mathbf{x}^{(k)}$, consider the line search problem (ref. Appendix C for details)

$$\lambda_k = \min_{\lambda} \; \phi\left(\mathbf{x}^{(k)} + \lambda \mathbf{g}^{(k)}\right)$$

where

$$\mathbf{g}^{(k)} = -(\mathbf{A}\mathbf{x}^{(k)} - \mathbf{b}) \tag{132}$$

Solving the one dimensional optimization problem yields

$$\lambda_k = \frac{\mathbf{b}^T\mathbf{g}^{(k)}}{\left(\mathbf{g}^{(k)}\right)^T \mathbf{A}\mathbf{g}^{(k)}}$$

Thus, the gradient search method can be summarized in Table (5).

Table 5: Gradient Search Method for Solving Linear Algebraic Equations

INITIALIZE: $\mathbf{x}^{(0)}, \varepsilon, k_{\max}, \lambda^{(0)}, \delta$

$k = 0$

$\mathbf{g}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$

WHILE $[(\delta > \varepsilon)$ **AND** $(k < k_{\max})]$

$$\lambda_k = -\frac{\mathbf{b}^T \mathbf{g}^{(k)}}{\left(\mathbf{g}^{(k)}\right)^T \mathbf{A}\mathbf{g}^{(k)}}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \, \mathbf{g}^{(k)}$$

$$\mathbf{g}^{(k+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)}$$

$$\delta = \frac{\left\|\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}\right\|_2}{\left\|\mathbf{g}^{(k+1)}\right\|_2}$$

$$\mathbf{g}^{(k)} = \mathbf{g}^{(k+1)}$$

$$k = k + 1$$

END WHILE

6.2 Conjugate Gradient Method

The gradient method makes fast progress initially when the guess is away from the optimum. However, this method tends to slow down as iterations progress. It can be shown that directions there exist better descent directions than the negative of the gradient direction. One such approach is conjugate gradient method. In conjugate directions method, we take search directions $\{\mathbf{s}^{(k)} : k = 0, 1, 2, ...\}$ such that they are orthogonal with respect to matrix A, i.e.

$$\left[\mathbf{s}^{(k)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)} = 0 \quad \text{for all } k \tag{133}$$

Such directions are called A-conjugate directions. To see how these directions are constructed, consider recursive scheme

$$\mathbf{s}^{(k)} = \beta_k \mathbf{s}^{(k-1)} + \mathbf{g}^{(k)} \tag{134}$$

$$\mathbf{s}^{(k-1)} = \overline{\mathbf{0}} \tag{135}$$

Premultiplying $\left[\mathbf{s}^{(k)}\right]^T$ by $\mathbf{A}\mathbf{s}^{(k-1)}$, we have

$$\left[\mathbf{s}^{(k)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)} = \beta_k \left[\mathbf{s}^{(k-1)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)} + \left[\mathbf{g}^{(k)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)} \tag{136}$$

A-conjugacy of directions $\{\mathbf{s}^{(k)} : k = 0, 1, 2, ...\}$ can be achieved if we choose

$$\beta_k = -\frac{\left[\mathbf{g}^{(k)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}}{\left[\mathbf{s}^{(k-1)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}} \tag{137}$$

Thus, given a search direction $\mathbf{s}^{(k-1)}$, new search direction is constructed as follows

$$\begin{aligned}\mathbf{s}^{(k)} &= -\frac{\left[\mathbf{g}^{(k)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}}{\left[\mathbf{s}^{(k-1)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}}\mathbf{s}^{(k-1)} + \mathbf{g}^{(k)} \\ &= \mathbf{g}^{(k)} - \left\langle \mathbf{g}^{(k)}, \widehat{\mathbf{s}}^{(k-1)} \right\rangle_A \widehat{\mathbf{s}}^{(k-1)} \end{aligned} \tag{138}$$

where

$$\widehat{\mathbf{s}}^{(k-1)} = \frac{\mathbf{s}^{(k-1)}}{\sqrt{\left[\mathbf{s}^{(k-1)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}}} = \frac{\mathbf{s}^{(k-1)}}{\sqrt{\left\langle \mathbf{s}^{(k-1)}, \mathbf{s}^{(k-1)} \right\rangle_A}}$$

It may be that matrix $\mathbf{A}$ is assumed to be symmetric and positive definite. Now, recursive use of equations (134-135) yields

$$\begin{aligned}\mathbf{s}^{(0)} &= \mathbf{g}^{(0)} \\ \mathbf{s}^{(1)} &= \beta_1 \mathbf{s}^{(0)} + \mathbf{g}^{(1)} = \beta_1 \mathbf{g}^{(0)} + \mathbf{g}^{(1)} \\ \mathbf{s}^{(2)} &= \beta_2 \mathbf{s}^{(1)} + \mathbf{g}^{(2)} \\ &= \beta_2\beta_1 \mathbf{g}^{(0)} + \beta_2 \mathbf{g}^{(1)} + \mathbf{g}^{(2)} \\ .... &= .... \\ \mathbf{s}^{(n)} &= (\beta_n ... \beta_1)\mathbf{g}^{(0)} + (\beta_n ... \beta_2)\mathbf{g}^{(1)} + ... + \mathbf{g}^{(n)}\end{aligned}$$

Thus, this procedure sets up each new search direction as a linear combination of all previous search directions and newly determined gradient.

Now, given the new search direction, the line search is formulated as follows

$$\lambda_k = \min_{\lambda} \phi\left(\mathbf{x}^{(k)} + \lambda \mathbf{s}^{(k)}\right) \tag{139}$$

Solving the one dimensional optimization problem yields

$$\lambda_k = \frac{\mathbf{b}^T \mathbf{s}^{(k)}}{\left(\mathbf{s}^{(k)}\right)^T \mathbf{A}\mathbf{s}^{(k)}} \tag{140}$$

Thus, the conjugate gradient search algorithm for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ is summarized in Table (6).

If conjugate gradient method is used for solving the optimization problem, it can be theoretically shown that the minimum can be reached in n steps. In practice, however, we require more than n steps to achieve $\phi(\mathbf{x}) < \varepsilon$ due to the rounding off errors in computation of the conjugate directions. Nevertheless, when n is large, this approach can generate a reasonably accurate solution with considerably less computations.

Table 6: Conjugate Gradient Algorithm to Solve Linear Algebraic Equations

INITIALIZE: $\mathbf{x}^{(0)}, \varepsilon, k_{\max}, \lambda^{(0)}, \delta$

$k = 0$

$\mathbf{g}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$

$\mathbf{s}^{(-1)} = \overline{\mathbf{0}}$

WHILE $[(\delta > \varepsilon)$ **AND** $(k < k_{\max})]$

$$\beta_k = -\frac{\left[\mathbf{g}^{(k)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}}{\left[\mathbf{s}^{(k-1)}\right]^T \mathbf{A}\mathbf{s}^{(k-1)}}$$

$$\mathbf{s}^{(k)} = \beta_k \mathbf{s}^{(k-1)} + \mathbf{g}^{(k)}$$

$$\lambda_k = -\frac{\mathbf{b}^T \mathbf{s}^{(k)}}{\left(\mathbf{s}^{(k)}\right)^T \mathbf{A}\mathbf{s}^{(k)}}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \, \mathbf{s}^{(k)}$$

$$\mathbf{g}^{(k+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)}$$

$$\delta = \frac{\left\|\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}\right\|_2}{\left\|\mathbf{g}^{(k+1)}\right\|_2}$$

$k = k + 1$

END WHILE

7 Matrix Conditioning and Behavior of Solutions

One of the important issue in computing solutions of large dimensional linear system of equations is the round-off errors caused by the computer. Some matrices are *well conditioned* and the computations proceed smoothly while some are inherently *ill conditioned,* which imposes limitations on how accurately the system of equations can be solved using any computer or solution technique. We now introduce measures for assessing whether a given system of linear algebraic equations is inherently *ill conditioned* or *well conditioned.*

Normally any computer keeps a fixed number of significant digits. For example, consider a computer that keeps only first three significant digits. Then, adding

$$0.234 + 0.00231 \rightarrow 0.236$$

results in loss of smaller digits in the smaller number. When a computer can commits millions of such errors in a complex computation, the question is, how do these individual errors contribute to the final error in computing the solution? Suppose we solve for $\mathbf{A}\mathbf{x} = \mathbf{b}$ using LU decomposition, the elimination algorithm actually produce approximate factors $\mathbf{L}'$ and $\mathbf{U}'$. Thus, we end up solving the problem with a wrong matrix, i.e.

$$\mathbf{A} + \delta\mathbf{A} = \mathbf{L}'\mathbf{U}' \tag{141}$$

instead of right matrix $\mathbf{A} = \mathbf{LU}$. In fact, due to round off errors inherent in any computation using computer, we actually end up solving the equation

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x}+\delta\mathbf{x}) = \mathbf{b}+\delta\mathbf{b} \tag{142}$$

The question is, how serious are the errors $\delta\mathbf{x}$ in solution $\mathbf{x}$, due to round off errors in matrix $\mathbf{A}$ and vector $\mathbf{b}$? Can these errors be avoided by rearranging computations or are the computations inherent ill-conditioned? In order to answer these questions, we need to develop some quantitative measure for **matrix conditioning**.

The following section provides motivation for developing a quantitative measure for matrix conditioning. In order to develop such a index, we need to define the concept of norm of a $m \times n$ matrix. The formal definition of matrix condition number and methods for computing it are presented in the later sub-sections.

7.1 Motivation [3]

In many situations, if the system of equations under consideration is **numerically well conditioned,** then it is possible to deal with the menace of round off errors by re-arranging the computations. If the system of equations is inherently an **ill conditioned** system, then the rearrangement trick does not help. Let us try and understand this by considering two simple examples and a computer that keeps only three significant digits.

Consider the system (**System-1**)

$$\begin{bmatrix} 0.0001 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{143}$$

If we proceed with Gaussian elimination without maximal pivoting , then the first elimination step yields

$$\begin{bmatrix} 0.0001 & 1 \\ 0 & -9999 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -9998 \end{bmatrix} \tag{144}$$

and with back substitution this results in

$$x_2 = 0.999899 \tag{145}$$

which will be rounded off to

$$x_2 = 1 \tag{146}$$

in our computer which keeps only three significant digits. The solution then becomes

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} 0.0 & 1 \end{bmatrix} \tag{147}$$

However, using maximal pivoting strategy the equations can be rearranged as

$$\begin{bmatrix} 1 & 1 \\ 0.0001 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{148}$$

and the Gaussian elimination yields

$$\begin{bmatrix} 1 & 1 \\ 0 & 0.9999 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0.9998 \end{bmatrix} \tag{149}$$

and again due to three digit round off in our computer, the solution becomes

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Thus, when A is a well conditioned numerically and Gaussian elimination is employed, the main reason for blunders in calculations is wrong pivoting strategy. If maximum pivoting is used then natural resistance of the system of equations to *round-off errors* is no longer compromised.

Now, to understand difficulties associated with **ill conditioned** systems, consider another system (**System-2**)

$$\begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \tag{150}$$

By Gaussian elimination

$$\begin{bmatrix} 1 & 1 \\ 0 & 0.0001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \tag{151}$$

If we change R.H.S. of the system 2 by a small amount

$$\begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix} \tag{152}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0.0001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0.0001 \end{bmatrix} \Longrightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{153}$$

Note that change in the fifth digit of second element of vector **b** was amplified to change in the first digit of the solution. Here is another example of an illconditioned matrix [[2]]. Consider the following system

$$\mathbf{Ax} = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix} \tag{154}$$

whose exact solution is $\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$. Now, consider a slightly perturbed system

$$\left[\mathbf{A} + \begin{bmatrix} 0 & 0 & 0.1 & 0.2 \\ 0.08 & 0.04 & 0 & 0 \\ 0 & -0.02 & -0.11 & 0 \\ -0.01 & -0.01 & 0 & -0.02 \end{bmatrix} \right] \mathbf{x} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix} \tag{155}$$

This slight perturbation in $\mathbf{A}$ matrix changes the solution to

$$\mathbf{x} = \begin{bmatrix} -81 & 137 & -34 & 22 \end{bmatrix}^T$$

Alternatively, if vector $\mathbf{b}$ on the R.H.S. is changed to

$$b = \begin{bmatrix} 31.99 & 23.01 & 32.99 & 31.02 \end{bmatrix}^T$$

then the solution changes to

$$\mathbf{x} = \begin{bmatrix} 0.12 & 2.46 & 0.62 & 1.23 \end{bmatrix}^T$$

Thus, matrices A in System 2 and in equation (154) are **ill conditioned**. Hence, no numerical method can avoid sensitivity of these systems of equations to small permutations, which can result even from truncation errors. The ill conditioning can be shifted from one place to another but it cannot be eliminated.

7.2 Condition Number [3]

Condition number of a matrix is a measure to quantify matrix Ill-conditioning. Consider system of equations given as $\mathbf{Ax} = \mathbf{b}$. We examine two situations: (a) errors in representation of vector $\mathbf{b}$ and (b) errors in representation of matrix $\mathbf{A}$.

7.2.1 Case: Perturbations in vector b [3]

Consider the case when there is a change in $\mathbf{b}$,i.e., $\mathbf{b}$ changes to $\mathbf{b} + \delta\mathbf{b}$ in the process of numerical computations. Such an error may arise from experimental errors or from round off errors. This perturbation causes a change in solution from $\mathbf{x}$ to $\mathbf{x} + \delta\mathbf{x}$, i.e.

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \tag{156}$$

By subtracting $\mathbf{Ax} = \mathbf{b}$ from the above equation we have

$$\mathbf{A}\delta\mathbf{x} = \delta\mathbf{b} \tag{157}$$

To develop a measure for conditioning of matrix $\mathbf{A}$, we compare relative change/error in solution,i.e. $||\delta\mathbf{x}|| \,/\, ||\mathbf{x}||$ to relative change in $\mathbf{b}$,i.e. $||\delta\mathbf{b}|| \,/\, ||\mathbf{b}||$. To derive this relationship, we consider the following two inequalities

$$\delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b} \Rightarrow ||\delta\mathbf{x}|| \leq ||\mathbf{A}^{-1}||\;||\delta\mathbf{b}|| \tag{158}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow ||\mathbf{b}|| = ||\mathbf{A}\mathbf{x}|| \leq ||\mathbf{A}||\;||\mathbf{x}|| \tag{159}$$

which follow from the definition of induced matrix norm. Combining these inequalities, we can write

$$||\delta\mathbf{x}||\;||\mathbf{b}|| \leq ||\mathbf{A}^{-1}||\;||\mathbf{A}||\;||\mathbf{x}||\;||\delta\mathbf{b}|| \tag{160}$$

$$\Rightarrow \frac{||\delta\mathbf{x}||}{||\mathbf{x}||} \leq (||\mathbf{A}^{-1}||\;||\mathbf{A}||)\;\frac{||\delta\mathbf{b}||}{||\mathbf{b}||} \tag{161}$$

$$\Rightarrow \frac{||\delta\mathbf{x}||/||\mathbf{x}||}{||\delta\mathbf{b}||/||\mathbf{b}||} \leq ||\mathbf{A}^{-1}||\;||\mathbf{A}|| \tag{162}$$

It may be noted that the above inequality holds for any vectors $\mathbf{b}$ and $\delta\mathbf{b}$. The number

$$c(\mathbf{A}) = ||\mathbf{A}^{-1}||\;||\mathbf{A}|| \tag{163}$$

is called as **condition number** of matrix $\mathbf{A}$. The condition number gives an upper bound on the possible amplification of errors in $\mathbf{b}$ while computing the solution [3].

7.2.2 Case: Perturbation in matrix A[3]

Suppose ,instead of solving for $\mathbf{A}\mathbf{x} = \mathbf{b}$, due to truncation errors, we end up solving

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} \tag{164}$$

Then, by subtracting $\mathbf{A}\mathbf{x} = \mathbf{b}$ from the above equation we obtain

$$\mathbf{A}\delta\mathbf{x} + \delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \overline{\mathbf{0}} \tag{165}$$

$$\Rightarrow \delta\mathbf{x} = -\mathbf{A}^{-1}\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) \tag{166}$$

Taking norm on both the sides, we have

$$||\delta\mathbf{x}|| = ||\mathbf{A}^{-1}\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x})|| \tag{167}$$

$$||\delta\mathbf{x}|| \leq ||\mathbf{A}^{-1}||\;||\delta\mathbf{A}||\;||\mathbf{x} + \delta\mathbf{x}| \tag{168}$$

$$\frac{||\delta\mathbf{x}||}{||\mathbf{x} + \delta\mathbf{x}||} \leq (||\mathbf{A}^{-1}||\;||\mathbf{A}||)\;\frac{||\delta\mathbf{A}||}{||\mathbf{A}||} \tag{169}$$

$$\frac{||\delta\mathbf{x}||/||\mathbf{x} + \delta\mathbf{x}||}{||\delta\mathbf{A}||/||\mathbf{A}||} \leq c(\mathbf{A}) = ||\mathbf{A}^{-1}||\;||\mathbf{A}|| \tag{170}$$

Again,the condition number gives an upper bound on % change in solution to % error A.

In simple terms, the condition number of a matrix tells us how serious is the error in solution of $\mathbf{Ax} = \mathbf{b}$ due to the truncation or round off errors in a computer. These inequalities mean that round off error comes from two sources

- Inherent or natural sensitivity of the problem,which is measured by $c(\mathbf{A})$
- Actual errors $\delta\mathbf{b}$ *or* $\delta\mathbf{A}$.

It has been shown that the maximum pivoting strategy is adequate to keep $(\delta\mathbf{A})$ in control so that the whole burden of round off errors is carried by the condition number $c(\mathbf{A})$. If condition number is high (>1000), the system is ill conditioned and is more sensitive to round off errors. If condition number is low (<100) system is well conditioned and you should check your algorithm for possible source of errors.

7.2.3 Computations of condition number

Let λ_n denote the largest magnitude eigenvalue of matrix $\mathbf{A}$ and λ_1 denote the smallest magnitude eigen value of $\mathbf{A}$. Then, we know that

$$||\mathbf{A}||_2^2 = \rho(\mathbf{A}^T\mathbf{A}) = \lambda_n \tag{171}$$

Also,

$$||\mathbf{A}^{-1}||_2^2 = \rho[(\mathbf{A}^{-1})^T\mathbf{A}^{-1}] = \rho\left[(\mathbf{A}\mathbf{A}^T)^{-1}\right] \tag{172}$$

This follows from identity

$$\begin{aligned} (\mathbf{A}^{-1}\mathbf{A})^T &= \mathbf{I} \\ \mathbf{A}^T(\mathbf{A}^{-1})^T &= \mathbf{I} \\ (\mathbf{A}^T)^{-1} &= (\mathbf{A}^{-1})^T \end{aligned} \tag{173}$$

Now, if λ is eigenvalue of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{v}$ is the corresponding eigenvector, then

$$(\mathbf{A}^T\mathbf{A})\mathbf{v} = \lambda\mathbf{v} \tag{174}$$

$$\mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{v}) = \lambda(\mathbf{A}\mathbf{v}) \tag{175}$$

λ is also eigenvalue of $\mathbf{A}\mathbf{A}^T$ and $(\mathbf{A}\mathbf{v})$ is the corresponding eigenvector. Thus, we can write

$$||\mathbf{A}^{-1}||_2^2 = \rho\left[(\mathbf{A}\mathbf{A}^T)^{-1}\right] = \rho\left[(\mathbf{A}^T\mathbf{A})^{-1}\right] \tag{176}$$

Also, since $\mathbf{A}\mathbf{A}^T$ is a symmetric positive definite matrix, we can diagonalize it as

$$\mathbf{A}^T\mathbf{A} = \mathbf{\Psi}\mathbf{\Lambda}\mathbf{\Psi}^T \tag{177}$$

$$\Rightarrow (\mathbf{A}^T\mathbf{A})^{-1} = [\mathbf{\Psi\Lambda\Psi}^T]^{-1} = (\mathbf{\Psi}^T)^{-1}\left[\mathbf{\Lambda}^{-1}\right]\mathbf{\Psi}^{-1} = \mathbf{\Psi\Lambda}^{-1}\mathbf{\Psi}^T$$

as Ψ is a unitary matrix. Thus, if λ is eigen value of $\mathbf{A}^T\mathbf{A}$ then $1/\lambda$ is eigen value of $(\mathbf{A}^T\mathbf{A})^{-1}$. If $\lambda_1$smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ then $1/\lambda_1$ is largest magnitude eigenvalue of $\mathbf{A}^T\mathbf{A}$

$$\Rightarrow \rho[(\mathbf{A}^T\mathbf{A})^{-1}] = 1/\ \lambda_1$$

Thus, the condition number of matrix $\mathbf{A}$ can be computed using 2-norm as

$$c_2(\mathbf{A}) = ||\mathbf{A}||_2\ ||\mathbf{A}^{-1}||_2 = \ (\lambda_n/\lambda_1)^{1/2}$$

where λ_n and λ_1 are largest and smallest magnitude eigenvalues of $\mathbf{A}^T\mathbf{A}$.

The condition number can also be estimated using any other norm. For example, if we use $\infty - norm$, then

$$c_\infty(\mathbf{A}) = ||\mathbf{A}||_\infty\ ||\mathbf{A}^{-1}||_\infty$$

Estimation of condition number by this approach, however, requires computation of $\mathbf{A}^{-1}$, which can be unreliable if $\mathbf{A}$ is ill conditioned.

Example 10 *[8]Consider the Hilbert matrix discussed in the module Problem Discretization using Approximation Theory. These matrices, which arise in simple polynomial approximation are notoriously ill conditioned and $c(\mathbf{H}_n) \to \infty$ as $n \to \infty$. For example, consider*

$$\begin{aligned}
\mathbf{H}_3 &= \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} \quad ; \quad \mathbf{H}_3^{-1} = \begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix} \\
\|\mathbf{H}_3\|_1 &= \|\mathbf{H}_3\|_\infty = 11/6 \quad \textit{and} \quad \left\|\mathbf{H}_3^{-1}\right\|_1 = \left\|\mathbf{H}_3^{-1}\right\|_\infty = 408
\end{aligned}$$

Thus, condition number can be computed as $c_1(\mathbf{H}_3) = c_\infty(\mathbf{H}_3) = 748$. For $n = 6$, $c_1(\mathbf{H}_3) = c_\infty(\mathbf{H}_3) = 29 \times 10^6$, which is extremely bad.

Even for $n = 3$, the effects of rounding off can be quite serious. For, example, the solution of

$$\mathbf{H}_3\mathbf{x} = \begin{bmatrix} 11/6 \\ 13/12 \\ 47/60 \end{bmatrix}$$

is $\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$. If we round off the elements of $\mathbf{H}_3$ to three significant decimal digits, we obtain

$$\begin{bmatrix} 1 & 0.5 & 0.333 \\ 0.5 & 0.333 & 0.25 \\ 0.333 & 0.25 & 0.2 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1.83 \\ 1.08 \\ 0.783 \end{bmatrix}$$

then the solution changes to $\mathbf{x}+\boldsymbol{\delta}\mathbf{x}=\begin{bmatrix} 1.09 & 0.488 & 1.491 \end{bmatrix}^T$. *The relative perturbation in elements of matrix* $\mathbf{H}_3$ *does not exceed 0.3%. However, the solution changes by 50%! The main indicator of ill-conditioning is that the magnitudes of the pivots become very small when Gaussian elimination is used to solve the problem.*

Example 11 *Consider matrix*

$$\mathbf{A}=\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

This matrix is near singular with eigen values (computed using Scilab)

$$\lambda_1 = 16.117 \ ; \lambda_2 = -1.1168 \ ; \ \ \lambda_3 = -1.3\times 10^{-15}$$

has the condition number of $c_2(\mathbf{A}) = 3.8131\times 10^{16}$. *If we attempt to compute inverse of this matrix using Scilab, we get following result*

$$\mathbf{A}^{-1} = 10^{16}\times\begin{bmatrix} -0.4504 & 0.9007 & -0.4504 \\ 0.9007 & -1.8014 & 0.9007 \\ -0.4504 & 0.9007 & -0.4504 \end{bmatrix}$$

*with a warning: '*Matrix is close to singular or badly scaled.*' The difficulties in computing inverse of this matrix are apparent if we further compute product* $\mathbf{A}\times\mathbf{A}^{-1}$, *which yields*

$$\mathbf{A}\times\mathbf{A}^{-1} = \begin{bmatrix} 2 & 0 & 2 \\ 8 & 0 & 0 \\ 16 & 0 & 8 \end{bmatrix}$$

On the other hand, consider matrix

$$\mathbf{B} = 10^{-17}\times\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

with eigenvalues

$$\lambda_1 = 4.73\times 10^{-17} \ ; \lambda_2 = \ -1\times 10^{-17} \ ; \ \ \lambda_3 = 1.26\times 10^{-17}$$

The eigenvalues are 'close to zero' the matrix is almost like a null matrix. However, the condition number of this matrix is $c_2(\mathbf{B}) = 5.474$. *If we proceed to compute of* $\mathbf{B}^{-1}$ *using Scilab, we get*

$$\mathbf{B}^{-1} = 10^{16}\times\begin{bmatrix} -1.67 & 8.33 & -5 \\ 6.67 & -3.33 & 0 \\ -1.67 & -1.67 & 5 \end{bmatrix}$$

and $\mathbf{B}\times\mathbf{B}^{-1}$*yields* $\mathbf{I}$, *i.e. identity matrix.*

Thus, it is important to realize that each system of linear equations has a inherent character, which can be quantified using the condition number of the associated matrix. The best of the linear equation solvers cannot overcome the computational difficulties posed by inherent ill conditioning of a matrix. As a consequence, when such ill conditioned matrices are encountered, the results obtained using any computer or any solver are unreliable.

8 Summary

In these lecture notes, we have developed methods for efficiently solving large dimensional linear algebraic equations. To begin with, we discuss geometric conditions for existence of the solutions. The direct methods for dealing with sparse matrices are discussed next. Iterative solution schemes and their convergence characteristics are discussed in the subsequent section. The concept of matrix condition number is then introduced to analyze susceptibility of a matrix to the round-off errors.

9 Appendix A: Behavior of Solutions of Linear Difference Equations

Consider difference equation of the form

$$\mathbf{z}^{(k+1)} = \mathbf{B}\mathbf{z}^{(k)} \tag{178}$$

where $\mathbf{z} \in \mathbf{R}^n$ and $\mathbf{B}$ is a $n \times n$ matrix. Starting from an initial condition $\mathbf{z}^{(0)}$, we get a sequence of vectors $\left\{\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, ..., \mathbf{z}^{(k)}, ...\right\}$ such that

$$\mathbf{z}^{(k)} = \mathbf{B}^k \mathbf{z}^{(0)}$$

for any k. Equations of this type are frequently encountered in numerical analysis. We would like to analyze asymptotic behavior of equations of these type without solving them explicitly.

To begin with, let us consider scalar linear iteration scheme

$$z^{(k+1)} = \beta z^{(k)} \tag{179}$$

where $z^{(k)} \in R$ and β is a real scalar. It can be seen that

$$\mathbf{z}^{(k)} = (\beta)^k \mathbf{z}^{(0)} \to 0 \text{ as } k \to \infty \tag{180}$$

if and only if $|\beta| < 1$.To generalize this notation to a multidimensional case, consider equation of type (178) where $\mathbf{z}^{(k)} \in R^n$. Taking motivation from the scalar case, we propose a solution to equation (178) of type

$$\mathbf{z}^{(k)} = \lambda^k \mathbf{v} \tag{181}$$

where λ is a scalar and $\mathbf{v} \in R^n$ is a vector. Substituting equation (181) in equation (178), we get

$$\begin{aligned} \lambda^{k+1}\mathbf{v} &= \mathbf{B}(\lambda^k \mathbf{v}) && (182) \\ \text{or } \lambda^k (\lambda I - \mathbf{B})\,\mathbf{v} &= \overline{0} && (183) \end{aligned}$$

Since we are interested in a non-trivial solution, the above equation can be reduced to

$$(\lambda I - \mathbf{B})\,\mathbf{v} = \overline{0} \tag{184}$$

where $\mathbf{v} \neq \overline{0}$. Note that the above set of equations has n equations in $(n+1)$ unknowns (λ and n elements of vector $\mathbf{v}$). Moreover, these equations are nonlinear. Thus, we need to generate an additional equation to be able to solve the above set exactly. Now, the above equation can hold only when the columns of matrix $(\lambda I - \mathbf{B})$ are linearly dependent and $\mathbf{v}$ belongs to null space of $(\lambda I - \mathbf{B})$. If columns of matrix $(\lambda I - \mathbf{B})$ are linearly dependent, matrix $(\lambda I - \mathbf{B})$ is singular and we have

$$\det(\lambda I - \mathbf{B}) = 0 \tag{185}$$

Note that equation (185) is nothing but the characteristic polynomial of matrix A and its roots are called eigenvalues of matrix A. For each eigenvalue λ_i we can find the corresponding eigen vector $\mathbf{v}^{(i)}$ such that

$$\mathbf{B}\mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)} \tag{186}$$

Thus, we get n fundamental solutions of the form $(\lambda_i)^k \mathbf{v}^{(i)}$ to equation (178) and a general solution to equation (178) can be expressed as linear combination of these fundamental solutions

$$\mathbf{z}^{(k)} = \alpha_1 (\lambda_1)^k \mathbf{v}^{(1)} + \alpha_2 (\lambda_2)^k \mathbf{v}^{(2)} + ... + \alpha_n (\lambda_n)^k \mathbf{v}^{(n)} \tag{187}$$

Now, at $k = 0$ this solution must satisfy the condition

$$\begin{aligned} \mathbf{z}^{(0)} &= \alpha_1 \mathbf{v}^{(1)} + (\alpha_2)^k \mathbf{v}^{(2)} + ... + \alpha_n \mathbf{v}^{(n)} && (188) \\ &= \begin{bmatrix} \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & .... & \mathbf{v}^{(n)} \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & .... & \alpha_n \end{bmatrix}^T && (189) \\ &= \Psi \boldsymbol{\alpha} && (190) \end{aligned}$$

where Ψ is a $n \times n$ matrix with eigenvectors as columns and $\boldsymbol{\alpha}$ is a $n \times 1$ vector of n coefficients. Let us consider the special case when the eigenvectors are linearly independent. Then, we can express $\boldsymbol{\alpha}$ as

$$\boldsymbol{\alpha} = \Psi^{-1}\mathbf{z}^{(0)} \tag{191}$$

Behavior of equation (187) can be analyzed as $k \to \infty$.Contribution due to the i'th fundamental solution $(\lambda_i)^k \mathbf{v}^{(i)} \to \overline{0}$ if and only if $|\lambda_i| < 1$.Thus, $\mathbf{z}^{(k)} \to \overline{0}$ as $k \to \infty$ if and only if

$$|\lambda_i| < 1 \text{ for } i = 1, 2, ...n \tag{192}$$

If we define **spectral radius** of matrix A as

$$\rho(\mathbf{B}) = \max_i |\lambda_i| \tag{193}$$

then, the condition for convergence of iteration equation (178) can be stated as

$$\rho(\mathbf{B}) < 1 \tag{194}$$

Equation (187) can be further simplified as

$$\mathbf{z}^{(k)} = \begin{bmatrix} \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & .... & \mathbf{v}^{(n)} \end{bmatrix} \begin{bmatrix} (\lambda_1)^k & 0 & ..... & 0 \\ 0 & (\lambda_2)^k & 0 & ... \\ .... & .... & ..... & .... \\ 0 & .... & 0 & (\lambda_n)^k \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_n \end{bmatrix} \tag{195}$$

$$= \boldsymbol{\Psi} \begin{bmatrix} (\lambda_1)^k & 0 & ..... & 0 \\ 0 & (\lambda_2)^k & 0 & ... \\ .... & .... & ..... & .... \\ 0 & .... & 0 & (\lambda_n)^k \end{bmatrix} \boldsymbol{\Psi}^{-1}\mathbf{z}^{(0)} = \boldsymbol{\Psi}\left(\boldsymbol{\Lambda}\right)^k \boldsymbol{\Psi}^{-1}\mathbf{z}^{(0)} \tag{196}$$

where Λ is the diagonal matrix

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & ..... & 0 \\ 0 & \lambda_2 & 0 & ... \\ .... & .... & ..... & .... \\ 0 & .... & 0 & \lambda_n \end{bmatrix} \tag{197}$$

Now, consider set of n equations

$$\mathbf{B}\mathbf{v}^{(i)} = \lambda_i \mathbf{v}^{(i)} \quad \text{for} \quad (i = 1, 2, ...n) \tag{198}$$

which can be rearranged as

$$\begin{aligned} \boldsymbol{\Psi} &= \begin{bmatrix} \mathbf{v}^{(1)} & \mathbf{v}^{(2)} & .... & \mathbf{v}^{(n)} \end{bmatrix} \\ \mathbf{B}\boldsymbol{\Psi} &= \begin{bmatrix} \lambda_1 & 0 & ..... & 0 \\ 0 & \lambda_2 & 0 & ... \\ .... & .... & ..... & .... \\ 0 & .... & 0 & \lambda_n \end{bmatrix} \Psi \qquad (199) \\ \text{or } \mathbf{B} &= \boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^{-1} \qquad (200) \end{aligned}$$

Using above identity, it can be shown that

$$\mathbf{B}^k = \left(\boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^{-1}\right)^k = \boldsymbol{\Psi}\left(\boldsymbol{\Lambda}\right)^k \boldsymbol{\Psi}^{-1} \qquad (201)$$

and the solution of equation (178) reduces to

$$\mathbf{z}^{(k)} = \mathbf{B}^k \mathbf{z}^{(0)} \qquad (202)$$

and $\mathbf{z}^{(k)} \rightarrow \overline{0}$ as $k \rightarrow \infty$ if and only if $\rho(\mathbf{B}) < 1$. The largest magnitude eigen value, i.e., $\rho(\mathbf{B})$ will eventually dominate and determine the rate at which $\mathbf{z}^{(k)} \rightarrow \bar{0}$. The result proved in this section can be summarized as follows:

Theorem 12 *A sequence of vectors* $\left\{\mathbf{z}^{(k)} : k = 0, 1, , 2, ...\right\}$ *generated by the iteration scheme*

$$\mathbf{z}^{(k+1)} = \mathbf{B}\mathbf{z}^{(k)}$$

where $\mathbf{z} \in R^n$ *and* $\mathbf{B} \in R^n \times R^n$, *starting from any arbitrary initial condition* $\mathbf{z}^{(0)}$ *will converge to limit* $\mathbf{z}^* = \bar{0}$ *if and only if*

$$\rho(\mathbf{B}) < 1$$

Note that computation of eigenvalues is a computationally intensive task. The following theorem helps in deriving a sufficient conditions for convergence of linear iterative equations.

Theorem 13 *For a* $n \times n$ *matrix* $\mathbf{B}$, *the following inequality holds for any induced matrix norm*

$$\rho(\mathbf{B}) \leq \|\mathbf{B}\| \qquad (203)$$

Proof. Let λ_i be eigen value of $\mathbf{B}$ and $\mathbf{v}^{(i)}$ be the corresponding eigenvector. Then, we can write

$$\left\|\mathbf{B}\mathbf{v}^{(i)}\right\| = \left\|\lambda_i \mathbf{v}^{(i)}\right\| = |\lambda_i| \left\|\mathbf{v}^{(i)}\right\| \qquad (204)$$

for $i = 1, 2, ..., n$. From these equations, it follows that

$$\rho(\mathbf{B}) = \max_i |\lambda_i| = \max_i \frac{\left\|\mathbf{B}\mathbf{v}^{(i)}\right\|}{\|\mathbf{v}^{(i)}\|} \tag{205}$$

Using the definition of the induced matrix norm, we have

$$\frac{\|\mathbf{B}\mathbf{z}\|}{\|\mathbf{z}\|} \leq \|\mathbf{B}\| \tag{206}$$

for any $\mathbf{z} \in R^n$. Thus, it follows that

$$\rho(\mathbf{B}) = \max_i \frac{\left\|\mathbf{B}\mathbf{v}^{(i)}\right\|}{\|\mathbf{v}^{(i)}\|} \leq \|\mathbf{B}\| \tag{207}$$

■

Since $\rho(\mathbf{B}) \leq \|\mathbf{B}\| < 1 \Rightarrow \rho(\mathbf{B}) < 1$, a sufficient condition for convergence of iterative scheme can be derived as follows

$$\|\mathbf{B}\| < 1 \tag{208}$$

The above sufficient condition is more useful from the viewpoint of computations as $\|\mathbf{B}\|_1$ and $\|\mathbf{B}\|_\infty$ can be computed quite easily. On the other hand, the spectral radius of a large matrix can be comparatively difficult to compute.

10 Appendix B: Theorems on Convergence of Iterative Schemes

Theorem 2 [2]: *A sufficient condition for the convergence of Jacobi and Gauss-Seidel methods is that the matrix A of linear system $Ax = b$ is strictly diagonally dominant.*

Proof. For Jacobi method, we have

$$\mathbf{S}^{-1}\mathbf{T} = -\mathbf{D}^{-1}[\mathbf{L} + \mathbf{U}] \tag{209}$$

$$= \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & ..... & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{12}}{a_{22}} & 0 & ..... & .... \\ ..... & ..... & .... & -\frac{a_{n-1,n}}{a_{n-1,n-1}} \\ -\frac{a_{12}}{a_{nn}} & ..... & ..... & 0 \end{bmatrix} \tag{210}$$

■

As matrix A is diagonally dominant, we have

$$\sum_{j=1(j\neq i)}^{n} |a_{ij}| \quad < \quad |a_{ii}| \quad \text{for} \;\; i = 1, 2., ...n \tag{211}$$

$$\Rightarrow \quad \sum_{j=1(j\neq i)}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \quad \text{for} \;\; i = 1, 2., ...n \tag{212}$$

$$\left\|\mathbf{S}^{-1}\mathbf{T}\right\|_{\infty} \quad = \quad \max_{i} \left[\sum_{j=1(j\neq i)}^{n} \left|\frac{a_{ij}}{a_{ii}}\right|\right] < 1 \tag{213}$$

Thus, Jacobi iteration converges if A is diagonally dominant.

For Gauss Seidel iterations, the iteration equation for i'th component of the vector is given as

$$x_i^{(k+1)} = \left(\frac{1}{a_{ii}}\right)\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right] \tag{214}$$

Let $\mathbf{x}^*$ denote the true solution of $\mathbf{Ax} = \mathbf{b}$. Then, we have

$$x_i^* = \left(\frac{1}{a_{ii}}\right)\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^* - \sum_{j=i+1}^{n} a_{ij}x_j^*\right] \tag{215}$$

Subtracting (215) from (214), we have

$$x_i^{(k+1)} - x_i^* = \left(\frac{1}{a_{ii}}\right)\left[\sum_{j=1}^{i-1} a_{ij}\left(x_j^* - x_j^{(k+1)}\right) + \sum_{j=i+1}^{n} a_{ij}\left(x_j^* - x_j^{(k)}\right)\right] \tag{216}$$

or

$$\left|x_i^{(k+1)} - x_i^*\right| \leq \left[\sum_{j=1}^{i-1} \left|\frac{a_{ij}}{a_{ii}}\right| \left|\left(x_j^* - x_j^{(k+1)}\right)\right| + \sum_{j=i+1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \left|\left(x_j^* - x_j^{(k)}\right)\right|\right] \tag{217}$$

Since

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_{\infty} = \max_{j} \left|\left(x_j^* - x_j^{(k)}\right)\right|$$

we can write

$$\left|x_i^{(k+1)} - x_i^*\right| \leq p_i \left\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\right\|_{\infty} + q_i \left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_{\infty} \tag{218}$$

where

$$p_i = \sum_{j=1}^{i-1} \left|\frac{a_{ij}}{a_{ii}}\right| \quad ; \quad q_i = \sum_{j=i+1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \tag{219}$$

Let s be value of index i for which

$$\left|x_s^{(k+1)} - x_s^*\right| = \max_{j} \left|\left(x_j^* - x_j^{(k+1)}\right)\right| \tag{220}$$

Then, assuming $i = s$ in inequality (218), we get

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\right\|_\infty \leq p_i \left\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\right\|_\infty + q_i \left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_\infty \tag{221}$$

or

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\right\|_\infty \leq \frac{q_s}{1 - p_s} \left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_\infty \tag{222}$$

Let

$$\mu = \max_j \frac{q_j}{1 - p_j} \tag{223}$$

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\right\|_\infty \leq \frac{q_s}{1 - p_s} \left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_\infty \tag{224}$$

then it follows that

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\right\|_\infty \leq \mu \left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_\infty \tag{225}$$

Now, as matrix $\mathbf{A}$ is diagonally dominant, we have

$$0 < p_i < 1 \text{ and } 0 < q_i < 1 \tag{226}$$

$$0 < p_i + q_i = \sum_{j=1(j\neq i)}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \tag{227}$$

Let

$$\beta = \max_i \left[\sum_{j=1(j\neq i)}^{n} \left|\frac{a_{ij}}{a_{ii}}\right|\right] \tag{228}$$

Then, we have

$$p_i + q_i \leq \beta < 1 \tag{229}$$

It follows that

$$q_i \leq \beta - p_i \tag{230}$$

and

$$\mu = \frac{q_i}{1 - p_i} \leq \frac{\beta - p_i}{1 - p_i} \leq \frac{\beta - p_i\beta}{1 - p_i} = \beta < 1 \tag{231}$$

Thus, it follows from inequality (225) that

$$\left\|\mathbf{x}^* - \mathbf{x}^{(k)}\right\|_\infty \leq \mu^k \left\|\mathbf{x}^* - \mathbf{x}^{(0)}\right\|_\infty$$

i.e. the iteration scheme is a contraction map and

$$\lim_{k \to \infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

Theorem 3 [5]: *The Gauss-Seidel iterations converge if matrix A is symmetric and positive definite.*

Proof. For Gauss-Seidel method, when matrix $\mathbf{A}$ is symmetric, we have

$$\mathbf{S}^{-1}\mathbf{T} = (\mathbf{L}+\mathbf{D})^{-1}(-\mathbf{U}) = -(\mathbf{L}+\mathbf{D})^{-1}(\mathbf{L}^T)$$

Now, let $\mathbf{e}$ represent unit eigenvector of matrix $\mathbf{S}^{-1}\mathbf{T}$ corresponding to eigenvalue λ, i.e.

$$\begin{aligned} -(\mathbf{L}+\mathbf{D})^{-1}(\mathbf{L}^T)\mathbf{e} &= \lambda\mathbf{e} \\ \text{or} \qquad \mathbf{L}^T\mathbf{e} &= -\lambda(\mathbf{L}+\mathbf{D})\mathbf{e} \end{aligned}$$

Taking inner product of both sides with $\mathbf{e}$,we have

$$\begin{aligned} \left\langle \mathbf{L}^T\mathbf{e}, \mathbf{e} \right\rangle &= -\lambda \left\langle (\mathbf{L}+\mathbf{D})\mathbf{e}, \mathbf{e} \right\rangle \\ \lambda &= -\frac{\left\langle \mathbf{L}^T\mathbf{e}, \mathbf{e} \right\rangle}{\langle \mathbf{De}, \mathbf{e} \rangle + \langle \mathbf{Le}, \mathbf{e} \rangle} = -\frac{\langle \mathbf{e}, \mathbf{Le} \rangle}{\langle \mathbf{De}, \mathbf{e} \rangle + \langle \mathbf{Le}, \mathbf{e} \rangle} \end{aligned}$$

Defining

$$\begin{aligned} \alpha &= \langle \mathbf{Le}, \mathbf{e} \rangle = \langle \mathbf{e}, \mathbf{Le} \rangle \\ \sigma &= \langle \mathbf{De}, \mathbf{e} \rangle = \sum_{i=1}^{n} a_{ii} \left(e_i\right)^2 > 0 \end{aligned}$$

we have

$$\lambda = -\frac{\alpha}{\alpha+\sigma} \Rightarrow |\lambda| = \left| \frac{\alpha}{\alpha+\sigma} \right|$$

Note that $\sigma > 0$ follows from the fact that trace of matrix $\mathbf{A}$, is positive as eigenvalues of $\mathbf{A}$ are positive. Using positive definiteness of matrix $\mathbf{A}$, we have

$$\begin{aligned} \langle \mathbf{Ae}, \mathbf{e} \rangle &= \langle \mathbf{Le}, \mathbf{e} \rangle + \langle \mathbf{De}, \mathbf{e} \rangle + \left\langle \mathbf{L}^T\mathbf{e}, \mathbf{e} \right\rangle \\ &= \sigma + 2\alpha > 0 \end{aligned}$$

This implies

$$-\alpha < (\sigma + \alpha)$$

Since $\sigma > 0$, we can say that

$$\alpha < (\sigma + \alpha)$$

i.e.

$$|\alpha| < (\sigma + \alpha)$$

This implies

$$|\lambda| = \left| \frac{\alpha}{\alpha+\sigma} \right| < 1$$

■

Theorem 4 [3]: *For an arbitrary matrix A, the necessary condition for the convergence of relaxation method is* $0 < \omega < 2$.

Proof. The relaxation iteration equation can be given as

$$\mathbf{x}^{(k+1)} = (\mathbf{D} + \omega\mathbf{L})^{-1}\left[[(1-\omega)\mathbf{D} - \omega\mathbf{U}]\mathbf{x}^{(k)} + \omega\,\mathbf{b}\right] \tag{232}$$

Defining

$$\mathbf{B}_\omega = (\mathbf{D} + \omega\mathbf{L})^{-1}\left[(1-\omega)\mathbf{D} - \omega\mathbf{U}\right] \tag{233}$$

$$\det(\mathbf{B}_\omega) = \det\left[(\mathbf{D} + \omega\mathbf{L})^{-1}\right]\det\left[(1-\omega)\mathbf{D} - \omega\mathbf{U}\right] \tag{234}$$

Now, using the fact that the determinant of a triangular matrix is equal to multiplication of its diagonal elements, we have

$$\det(\mathbf{B}_\omega) = \det\left[\mathbf{D}^{-1}\right]\det\left[(1-\omega)\mathbf{D}\right] = (1-\omega)^n \tag{235}$$

Using the result that product of eigenvalues of $\mathbf{B}_\omega$ is equal to determinant of $\mathbf{B}_\omega$,we have

$$\lambda_1\lambda_2...\lambda_n = (1-\omega)^n \tag{236}$$

where $\lambda_i\,(i = 1, 2...n)$ denote eigenvalues of $\mathbf{B}_\omega$.

$$|\lambda_1\lambda_2...\lambda_n| = |\lambda_1|\,|\lambda_2|\,....\,|\lambda_n| = |(1-\omega)^n| \tag{237}$$

It is assumed that iterations converge. Now, convergence criterion requires

$$\lambda_i(\mathbf{B}_\omega) < 1 \quad \text{for } i = 1, 2, ...n \tag{238}$$

$$\Rightarrow \quad |\lambda_1|\,|\lambda_2|\,....\,|\lambda_n| < 1 \tag{239}$$

$$\Rightarrow \quad |(1-\omega)^n| < 1 \tag{240}$$

This is possible only if

$$0 < \omega < 2 \tag{241}$$

The optimal or the best choice of the ω is the one that makes spectral radius $\rho(\mathbf{B}_\omega)$smallest possible and gives fastest rate of convergence. ■

11 Appendix C: Steepest Descent / Gradient Search Method

In the module on *Problem Discretization using Approximation Theory*, we derived the necessary condition for a point $\mathbf{x} = \overline{\mathbf{x}}$ to be an extreme point of a twice differentiable scalar function $\phi(\mathbf{x}) : R^n \rightarrow R$ and sufficient conditions for the extreme point to qualify either as a minimum or a maximum. The question that needs to be answered in practice is, given $\phi(\mathbf{x})$ how does one locate an extreme point $\overline{\mathbf{x}}$. A direct approach can be to solve for n simultaneous equations

$$\nabla\phi(\mathbf{x}) = \overline{\mathbf{0}} \tag{242}$$

in n unknowns. When these equations are bell behaved nonlinear functions of $\mathbf{x}$, an iterative scheme, such as Newton-Raphson method, can be employed. Alternatively, iterative search schemes can be derived using evaluations of $\phi(\mathbf{x})$ and its derivatives. The steepest descent / gradient method is the simplest iterative search scheme in the unconstrained optimization and forms a basis for developing many sophisticated optimization algorithms. In this section, we discuss details of this numerical optimization approach.

11.1 Gradient / Steepest Descent / Cauchy's method

Set of vectors $\mathbf{x} \in R^{\mathbf{N}}$such that $\phi(\mathbf{x}) = \alpha$ where α is a constant, is called the level surface of $\phi(\mathbf{x})$ for value α. By tracing out one by one level surfaces we obtain contour plot (see Figure 2). Suppose $\mathbf{x} = \mathbf{x}^{(k)}$ is a point lying on one of the level surfaces. If $\phi(\mathbf{x})$ is continuous and differentiable then, using Taylor series expansion in a neighborhood of $\mathbf{x}^{(k)}$ we can write

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^{(k)}) + \left[\nabla\phi(\mathbf{x}^{(k)})\right]^T (\mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)})^T \left[\nabla^2\phi(\mathbf{x}^{(k)})\right] (\mathbf{x} - \mathbf{x}^{(k)}) + ... \tag{243}$$

If we neglect the second and higher order terms, we obtained

$$\begin{aligned} \phi(\mathbf{x}) &= \phi(\mathbf{x}^{(k)} + \Delta\mathbf{x}) \simeq \phi(\mathbf{x}^{(k)}) + \left[\nabla\phi(\mathbf{x}^{(k)})\right]^T \Delta\mathbf{x} = C & (244) \\ \Delta\mathbf{x} &= (\mathbf{x} - \mathbf{x}^{(k)}) & (245) \end{aligned}$$

This is equation of the plane tangent to surface $\phi(\mathbf{x})$ at point $\mathbf{x}^{(k)}$. The equation of level surface through $\mathbf{x}^{(k)}$ is

$$C = \phi(\mathbf{x}) = \phi(\mathbf{x}^{(k)}) \tag{246}$$

Combining above two equations, the equation of tangent surface at $\mathbf{x} = \mathbf{x}^{(k)}$ is obtained as

$$(\mathbf{x} - \mathbf{x}^{(k)})^T \nabla\phi(\mathbf{x}^{(k)}) = 0 \tag{247}$$

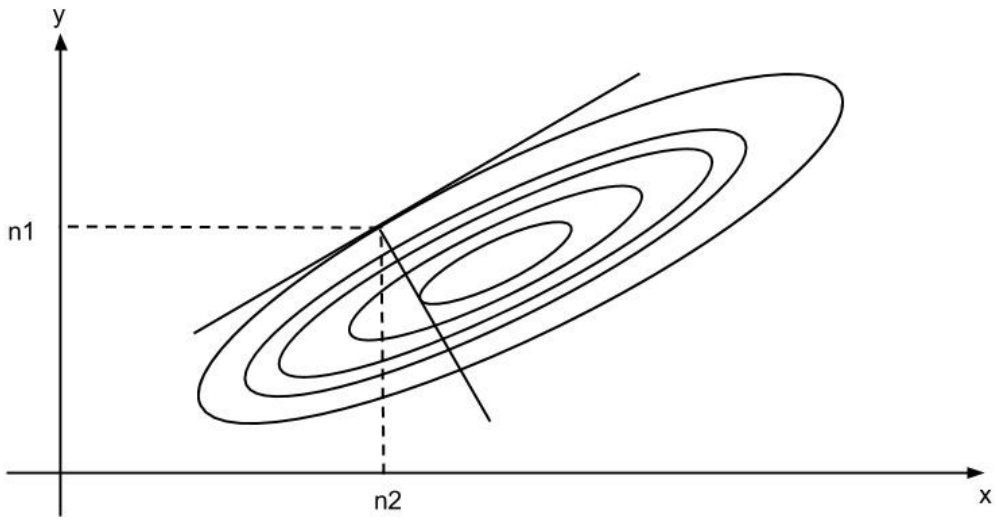

Figure 2: Contour plots (level surfaces) of $y = \phi(\mathbf{x})$ and the local tangent and the steepest descent direction at point (n2,n1)

Thus, gradient at $\mathbf{x} = \mathbf{x}^{(k)}$ is perpendicular to the level surface passing through $\phi(\mathbf{x}^{(k)})$ (See Figure 2).

We will now show that it points in the direction in which the function increases most rapidly, and in fact, $\nabla\phi(\mathbf{x}^{(k)})$ is the direction of maximum slope. If

$$\left[\nabla\phi(\mathbf{x}^{(k)})\right]^T \Delta\mathbf{x} < 0$$

then

$$\phi(\mathbf{x}^{(k)} + \Delta\mathbf{x}) < \phi(\mathbf{x}^{(k)}) \tag{248}$$

and $\Delta\mathbf{x}$ is called as descent direction. Suppose we fix our selves to unit sphere in the neighborhood of $\mathbf{x} = \mathbf{x}^{(k)}$ i.e. set of all $\mathbf{x}$ such that $\|\Delta\mathbf{x}\| \leq 1$ and want to find direction $\Delta\mathbf{x}$ such that $\Delta\phi(\mathbf{x})^T\Delta\mathbf{x}$ algebraically minimum. Using Cauchy-Schwartz inequality together with $\|\Delta\mathbf{x}\| \leq 1$, we have

$$\left|\left[\nabla\phi(\mathbf{x}^{(k)})\right]^T \Delta\mathbf{x}\right| \leq \left\|\nabla\phi(\mathbf{x}^{(k)})\right\| \|\Delta\mathbf{x}\| \leq \left\|\nabla\phi(\mathbf{x}^{(k)})\right\| \tag{249}$$

This implies

$$-\left\|\nabla\phi(\mathbf{x}^{(k)})\right\| \leq \left[\nabla\phi(\mathbf{x}^{(k)})\right]^T \Delta\mathbf{x} \leq \left\|\nabla\phi(\mathbf{x}^{(k)})\right\| \tag{250}$$

and minimum value $\left[\nabla\phi(\mathbf{x}^{(k)})\right]^T \Delta\mathbf{x}$ can attain when $\Delta\mathbf{x}$ is restricted within the unit ball equals $-\left\|\nabla\phi(\mathbf{x}^{(k)})\right\|$. In fact, the equality will hold if and only if $\Delta\mathbf{x}$ is chosen colinear with $\nabla\phi(\mathbf{x}^{(k)})$. Let $\mathbf{g}^{(k)}$ denote unit vector along the -ve of the gradient direction, i.e.

$$\mathbf{g}^{(k)} = -\frac{\nabla\phi(\mathbf{x}^{(k)})}{\|\nabla\phi(\mathbf{x}^{(k)})\|} \tag{251}$$

Then, $\mathbf{g}^{(k)}$ is the direction of steepest or maximum descent in which $\phi(\mathbf{x}^{(k)} + \Delta\mathbf{x}) - \phi(\mathbf{x}^{(k)})$ reduces at the maximum rate. While these arguments provides us with the local direction for the steepest descent, they does not give any clue on how much we should move in that direction so that the function $\phi(\mathbf{x})$ continues to decresae. To arrive at an optimal step size, the subsequent guess is constructed as follows

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \mathbf{g}^{(k)} \tag{252}$$

and the step size λ_k is determined by solving the following one dimensional minimization (line search) problem

$$\lambda_k = \min_{\lambda} \phi\left(\mathbf{x}^{(k)} + \lambda\mathbf{g}^{(k)}\right)$$

There are several numerical approaches available for solving this one dimensional optimization problem (ref [9]). An approach based on cubic polynomial interpolation is presented in the next sub-section. The iterative gradient based search algorithm can be summarized as in Table 7.

Alternate criteria which can be used for termination of iterations are as follows

$$\frac{\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|}{\|\mathbf{x}^{(k+1)}\|} \leq \varepsilon_1$$

The Method of steepest descent may appear to be the best unconstrained minimization method. However, due to the fact that steepest descent is a local property, this method is not effective in many problems. If the objective function are distorted, then the method can be hopelessly slow.

11.2 Line Search: One Dimensional Optimization

In any multi-dimensional minimization problem, at each iteration, we have to solve a one dimensional minimization problem of the form

$$\min_{\lambda} \phi(\lambda) = \min_{\lambda} \phi\left(\mathbf{x}^{(k)} + \lambda\mathbf{s}^{(k)}\right) \tag{253}$$

Table 7: Gradient Search Algorithm

INITIALIZE: $\mathbf{x}^{(0)}, \varepsilon, k_{\max}, \lambda^{(0)}, \delta$

$k = 0$

$$\mathbf{g}^{(0)} = -\frac{\nabla\phi(\mathbf{x}^{(0)})}{\|\nabla\phi(\mathbf{x}^{(0)})\|}$$

WHILE $[(\delta > \varepsilon)$ **AND** $(k < k_{\max})]$

$$\lambda_k = \min_{\lambda} \phi\left(\mathbf{x}^{(k)} + \lambda\mathbf{g}^{(k)}\right)$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k\,\mathbf{g}^{(k)}$$

$$\mathbf{g}^{(k+1)} = -\frac{\nabla\phi(\mathbf{x}^{(k+1)})}{\|\nabla\phi(\mathbf{x}^{(k+1)})\|}$$

$$\delta = \frac{\left\|\nabla\phi(\mathbf{x}^{(k+1)}) - \nabla\phi(\mathbf{x}^{(k)})\right\|_2}{\|\nabla\phi(\mathbf{x}^{(k+1)})\|_2}$$

$k = k + 1$

END WHILE

where $\mathbf{s}^{(k)}$ is the descent (search) direction. (In gradient search method, $\mathbf{s}^{(k)} = \mathbf{g}^{(k)}$, i.e. -ve of the gradient direction.) There are many ways of solving this problem numerically ([9]). In this sub-section, we discuss the cubic interpolation method, which is one of the popular techniques for performing the line search.

The first step in the line search is to find bounds on the optimal step size λ^*. These are established by finding two points, say α and β, such that the slope $d\phi/d\lambda$

$$\frac{d\phi}{d\lambda} = \left[\frac{\partial\phi(\mathbf{x}^{(k)} + \lambda\mathbf{s}^{(k)})}{\partial(\mathbf{x}^{(k)} + \lambda\mathbf{s}^{(k)})}\right]^T \frac{\partial(\mathbf{x}^{(k)} + \lambda\mathbf{s}^{(k)})}{\partial\lambda} \tag{254}$$

$$= \left(\nabla\phi(\mathbf{x}^{(k)} + \lambda\mathbf{s}^{(k)})\right)^T \mathbf{s}^{(k)} \tag{255}$$

has opposite signs at these points. We know that at $\lambda = 0$,

$$\frac{d\phi(0)}{d\lambda} = \left(\nabla\phi(\mathbf{x}^{(k)})\right)^T \mathbf{s}^{(k)} < 0 \tag{256}$$

as $s^{(k)}$ is assumed to be a descent direction. Thus, we take α corresponding to $\lambda = 0$ and try to find a point $\lambda = \beta$ such that $d\phi/d\lambda > 0$. The point β can be taken as the first value out of $\lambda = h,\ 2h, 4h, 8h, \ldots$. for which $d\phi/d\lambda > 0$,where h is some pre-assigned initial step size. As $d\phi/d\lambda$ changes sign in the interval $[0, \beta]$, the optimum λ^* is bounded in the interval $[0, \beta]$.

The next step is to approximate $\phi(\lambda)$ over interval $[0, \beta]$ by a cubic interpolating polynomial for the form

$$\phi(\lambda) = a + b\lambda + c\lambda^2 + d\lambda^3 \tag{257}$$

The parameters a and b be computed as

$$\begin{aligned} \phi(0) &= a = \phi(\mathbf{x}^{(k)}) \\ \frac{d\phi(0)}{d\lambda} &= b = \left(\nabla\phi(\mathbf{x}^{(k)})\right)^T \mathbf{s}^{(k)} \end{aligned}$$

The parameters c and d can be computed by solving

$$\begin{aligned} \phi(\beta) &= a + b\beta + c\beta^2 + d\beta^3 \\ \frac{d\phi(\beta)}{d\lambda} &= b + 2c\beta + 3d\beta^2 \end{aligned}$$

i.e. by solving

$$\begin{bmatrix} \beta^2 & \beta^3 \\ 2\beta & 3\beta^2 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \phi\left(\mathbf{x}^{(k)} + \beta\mathbf{s}^{(k)}\right) - a - \beta b \\ \left(\mathbf{s}^{(k)}\right)^T \nabla\phi\left(\mathbf{x}^{(k)} + \beta\mathbf{s}^{(k)}\right) - b \end{bmatrix}$$

The application of necessary condition for optimality yields

$$\frac{d\phi}{d\lambda} = b + 2c\lambda + 3d\lambda^2 = 0 \tag{258}$$

i.e.

$$\lambda^* = \frac{-c \pm \sqrt{(c^2 - 3bd)}}{3d} \tag{259}$$

One of the two values correspond to the minimum. The sufficiency condition for minimum requires

$$\frac{d^2\phi}{d\lambda^2} = 2c + 6d\lambda^* > 0 \tag{260}$$

The fact that $d\phi/d\lambda$ has opposite sign at $\lambda = 0$ and $\lambda = \beta$ ensures that the equation 258 does not have imaginary roots.

Table 8: Line Search using Cubic Interpolation Algorithm

INITIALIZE: $\mathbf{x}^{(k)}, \mathbf{s}^{(k)}, h$
Step 1: Find β
    $\beta = h$
    WHILE $[d\phi(\beta)/d\lambda < 0]$
        $\beta = 2\beta$
    END WHILE
Step 2: Solve for a, b, c and d using $\mathbf{x}^{(k)}, \mathbf{s}^{(k)}$ and β
Step 3: Find λ^* using sufficient condition for optimality

Exercise

1. The true solution $\mathbf{A}x = b$ is slightly different from the elimination solution to $\mathbf{LU}x_0 = b$; $\mathbf{A} - \mathbf{LU}$ misses zero matrix because of round off errors. One possibility is to do everything in double precision, but a better and faster way is iterative refinement: Compute only one vector $r = b - \mathbf{A}x_0$ in double precision, solve $\mathbf{LU}y = r$,and add correction y to x_0 to generate an improved solution $x_1 = x_0 + y$.

   Problem:Multiply $x_1 = x_0 + y$, by $\mathbf{LU}$, write the result as splitting $Sx_1 = Tx_0 + b$, and explain why T is extremely small. This single step bring almost close to $\mathbf{x}$.

2. If A is orthonormal (unitary) matrix, show that $||\mathbf{A}|| = 1$ and also $c(\mathbf{A}) = 1$. Orthogonal matrices and their multiples($\alpha\mathbf{A}$) are the only perfectly conditioned matrices.

3. Find a vector x orthogonal to row space, and a vector y orthogonal to column space, of
$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 3 \\ 3 & 6 & 4 \end{bmatrix}$$

4. Show that vector $\mathbf{x} - \mathbf{y}$ is orthogonal to vector $\mathbf{x} + \mathbf{y}$ if and only if $||\mathbf{x}|| = ||\mathbf{y}||$.

5. For a positive definite matrix A, the Cholensky decomposition is $\mathbf{A} = \mathbf{L}\ \mathbf{D}\ \mathbf{L}^T = RR^T$ where $R = \mathbf{L}\mathbf{D}^{1/2}$. Show that the condition number of R is square root of condition number of A. It follows that Gaussian elimination needs no row exchanges for a positive definite matrix; the condition number does not deteriorate, since $c(\mathbf{A}) = c(R^T)c(R)$.

6. Show that for a positive definite symmetric matrix, the condition number can be obtained as
$$c(\mathbf{A}) = \lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A})$$

7. Prove the following inequalities/ identities

$$\|\mathbf{A} + B\| \leq \|\mathbf{A}\| + \|B\|$$

$$\|\mathbf{A}B\| \leq \|\mathbf{A}\| \, \|B\|$$

$$C(\mathbf{A}B) \leq C(\mathbf{A})C(B)$$

$$\|\mathbf{A}\|_2 = \left\|\mathbf{A}^T\right\|_2$$

8. Show that $\lambda_{\max}$, or even $max \ |\lambda_i|$, is not a satisfactory norm of a matrix, by finding a 2×2 counter examples to following inequalities

$$\lambda_{\max}(\mathbf{A} + B) \leq \lambda_{\max}(\mathbf{A}) + \lambda_{\max}(B)$$

$$\lambda_{\max}(\mathbf{A}B) \leq \lambda_{\max}(\mathbf{A})\lambda_{\max}(B)$$

9. For the positive definite matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

compute the condition number $C(\mathbf{A})$ and find the right hand side $\mathbf{b}$ of equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ and perturbation $\delta\mathbf{b}$ such that the error is worst possible, i.e.

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} = C(\mathbf{A})\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

10. A polynomial

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

passes through point (3, 2), (4, 3), (5, 4) and (6, 6) in an x-y coordinate system. Setup the system of equations and solve it for coefficients a_0 to a_3 by Gaussian elimination. The matrix in this example is (4 X 4) *Vandermonde matrix*. Larger matrices of this type tend to become ill-conditioned.

11. Solve using Gauss Jordan method.

$$u + v + w = -2$$

$$3u + 3v - w = 6$$

$$u - v + w = -1$$

to obtain $\mathbf{A}^{-1}$. What coefficient of v in the third equation, in place of present -1 , would make it impossible to proceed and force the elimination to break down?

12. Decide whether vector $\mathbf{b}$ belongs to column space spanned by $\mathbf{x}^{(1)}$,$\mathbf{x}^{(2)}$, ...

    (a) $\mathbf{x}^{(1)} = (1,1,0)$; $\mathbf{x}^{(2)} = (2,2,1)$; $\mathbf{x}^{(3)} = (0,2,0)$; $\mathbf{b} = (\mathbf{3.4.5})$

    (b) $\mathbf{x}^{(1)} = (1,2,0)$; $\mathbf{x}^{(2)} = (2,5,0)$; $\mathbf{x}^{(3)} = (0,0,2)$; $\mathbf{x}^{(4)} = (0,0,0)$; any $\mathbf{b}$

13. Find dimension and construct a basis for the four fundamental subspaces associated with each of the matrices.

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 & 4 & 0 \\ 0 & 2 & 8 & 0 \end{bmatrix} \quad ; \quad \mathbf{U}_2 = \begin{bmatrix} 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad ; \quad \mathbf{A}_3 = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \quad ; \quad \mathbf{U}_1 = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$$

14. Find a non -zero vector $\mathbf{x}^*$ orthogonal to all rows of

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 3 \\ 3 & 6 & 4 \end{bmatrix}$$

    (In other words, find Null space of matrix $\mathbf{A}$.) If such a vector exits, can you claim that the matrix is singular? Using above A matrix find one possible solution $\mathbf{x}$ for $\mathbf{Ax} = \mathbf{b}$ when $b = [\ 4 \quad 9 \quad 13\ ]^T$. Show that if vector x is a solution of the system $\mathbf{Ax} = \mathbf{b}$, then ($\mathbf{x}$+$\alpha\mathbf{x}^*$) is also a solution for any scalar α, i.e.

$$\mathbf{A}(\mathbf{x} + \alpha\mathbf{x}^*) = b$$

    Also, find dimensions of row space and column space of A.

15. If product of two matrices yields null matrix, i.e. $\mathbf{A}B = [0]$, show that column space of B is contained in null space of A and the row space of A is in the left null space of B.

16. Why there is no matrix whose row space and null space both contain the vector

$$\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$$

17. Find a 1×3 matrix whose null space consists of all vectors in R^3 such that $x_1 + 2x_2 + 4x_3 = 0$. Find a 3×3 matrix with same null space.

18. If V is a subspace spanned by

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} ; \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} ; \begin{bmatrix} 1 \\ 5 \\ 0 \end{bmatrix}$$

find matrix A that has V as its row space and matrix B that has V as its null space.

19. Find basis for each of subspaces and rank of matrix A

(a)

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 4 & 6 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} = \mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

20. Consider the following system

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{bmatrix}$$

Obtain $\mathbf{A}^{-1}$, $\det(\mathbf{A})$ and also solve for $[x_1\ x_2\ ]^T$. Obtain numerical values for $\varepsilon = 0.01$, 0.001 and 0.0001. See how sensitive is the solution to change in ε.

21. Consider system

$$\mathbf{A} = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} \quad ; \quad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

where A is Hilbert matrix with $a_{ij} = 1/(i+j-1)$, which is severely ill- conditioned. Solve using

(a) Gauss-Jordan elimination

(b) exact computations

(c) rounding off each number to 3 figures.

Perform 4 iterations each by

(a) Jacobi method

(b) Gauss- Seidel method

(c) Successive over-relaxation method with $\omega = 1.5$

Use initial guess $\mathbf{x}^{(0)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ and compare in each case how close to the $\mathbf{x}^{(4)}$ is to the exact solution. (Use 2-norm for comparison).

Analyze the convergence properties of the above three iterative processes using eigenvalues of the matrix $(S^{-1}T)$ in each case. Which iteration will converge to the true solution?

22. The Jacobi iteration for a general 2 by 2 matrix has

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad ; \quad \mathbf{D} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$$

If $\mathbf{A}$ is symmetric positive definite, find the eigenvalues of $J = S^{-1}T = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})$ and show

that Jacobi iterations converge.

23. It is desired to solve $\mathbf{A}x = b$ using Jacobi and Gauss-Seidel iteration scheme where

$$\mathbf{A} = \begin{bmatrix} 4 & 2 & 1 \\ 1 & 5 & 3 \\ 2 & 4 & 7 \end{bmatrix} \quad ; \quad \mathbf{A} = \begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} \quad ; \quad \mathbf{A} = \begin{bmatrix} -7 & 1 & -2 & 3 \\ 1 & 8 & 1 & 3 \\ -2 & 1 & -5 & 1 \\ 1 & 0 & -1 & -3 \end{bmatrix}$$

Will the Jacobi and Gauss-Seidel the iterations converge? Justify your answer. (Hint: Check for diagonal dominance before proceeding to compute eigenvalues).

24. Given matrix

$$J = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

find powers J^2, J^3 by direct multiplications. For which matrix $\mathbf{A}$ is this a Jacobi matrix $J = I - \mathbf{D}^{-1}\mathbf{A}$? Find eigenvalues of J.

25. The tridiagonal $n \times n$ matrix $\mathbf{A}$ that appears when finite difference method is used to solve second order PDE / ODE-BVP and the corresponding Jacobi matrix are as follows

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & ... & 0 \\ -1 & 2 & -1 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & -1 & 2 & -1 \\ 0 & ... & 0 & 1 & 2 \end{bmatrix} \quad ; \quad J = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & ... & 0 \\ 1 & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & 1 & 0 & 1 \\ 0 & ... & 0 & 1 & 0 \end{bmatrix}$$

Show that the vector

$$x = \begin{bmatrix} sin(\pi h) & sin(2\pi h) & ... & sin(nh) \end{bmatrix}^T$$

satisfies $J\mathbf{x} = \lambda\mathbf{x}$ with eigenvalue $\lambda = cos(\pi h)$. Here, $h = 1/(n+1)$ and hence $sin\left[(n+1)\pi h\right] = 0$.

Note: The other eigenvectors replace π by 2π, 3π,...., nπ. The other eigenvalues are $cos(2\pi h)$, $cos(3\pi h)$,..... , $cos(n\pi h)$ all smaller than $cos(\pi h) < 1$.

References

[1] Gupta, S. K.; Numerical Methods for Engineers. Wiley Eastern, New Delhi, 1995.

[2] Gourdin, A. and M Boumhrat; Applied Numerical Methods. Prentice Hall India, New Delhi.

[3] Strang, G.; Linear Algebra and Its Applications. Harcourt Brace Jevanovich College Publisher, New York, 1988.

[4] Kelley, C.T.; Iterative Methods for Linear and Nonlinear Equations, Philadelphia : SIAM, 1995.

[5] Demidovich, B. P. and I. A. Maron; Computational Mathematics. Mir Publishers, Moskow, 1976.

[6] Atkinson, K. E.; An Introduction to Numerical Analysis, John Wiley, 2001.

[7] Linfield, G. and J. Penny; Numerical Methods Using Matlab, Prentice Hall, 1999.

[8] Phillips, G. M. and P. J. Taylor; Theory and Applications of Numerical Analysis, Academic Press, 1996.

[9] Rao, S. S., Optimization: Theory and Applications, Wiley Eastern, New Delhi, 1978.

[10] Bazara, M.S., Sherali, H. D., Shetty, C. M., Nonlinear Programming, John Wiley, 1979.