# *Chapter 2*

# Linear System of Equations

## 2.1 Introduction

The *Joker* escaped because the *Batmobile* lost the wheel while chasing. To avoid the adverse publicity, *Barrel Motors* decided to provide *Batman* with their new and exclusive model *FruitBat NSX 2008*. Before setting out to chase the *Joker* around, *Batman* decided to explore his new car. To his surprise, he soon found out that the engine does not provide a steady acceleration in the initial period. From zero to a certain unknown speed $u$, the engine delivers an uneven acceleration. However, that point onwards, the engine provides a steady thrust and a constant acceleration of $f$. Let us denote the distance covered by the batmobile to reach a speed of $u$ from rest as $S_0$. Batman knew enough physics to write his distance of chase $S$ in terms of $f$, $u$, $S_0$ and time $t$ as:

$$S = S_0 + ut + \frac{1}{2} ft^2 \tag{2.1}$$

Since, each batmobile engine is custom manufactured, the $f$, $u$ and $S_0$ are engine specific constants. These constants need to be calibrated after every 1000 km. It is easy to see why it is so important for the Batman to calibrate these parameters precisely. Without these parameters, he will not be able to estimate the precise time his chase will take once his GPS system provides the distance $S$ of the target.

For the first calibration run, he located three cathedrals at distances of 439, 640 and 875m from his starting position. He timed the chases with a precision of 0.01 sec as 6.20, 14.20 and 22.40 seconds, respectively. So, the calibration equations become:

$$\begin{aligned} S_0 + 6.2u + 19.22f &= 439 \\ S_0 + 14.2u + 100.82f &= 640 \\ S_0 + 22.4u + 250.88f &= 875 \end{aligned} \tag{2.2}$$

In the matrix form, the set of equations can be written as:

$$\begin{bmatrix} 1 & 6.2 & 19.22 \\ 1 & 14.2 & 100.82 \\ 1 & 22.4 & 250.88 \end{bmatrix} \begin{bmatrix} S_0 \\ u \\ f \end{bmatrix} = \begin{bmatrix} 439 \\ 640 \\ 875 \end{bmatrix} \tag{2.3}$$

Batman needs to program the solution of this system of equations into his batmobile computer. For recalibration of the constants, he will then need to make three trial runs to pre-selected targets and measure the travel times. The computer can then take the distance and time data, re-evaluate the constants and store them for future use.

For an actual chase, the global positioning system (GPS) picks up the actual distance of the target object and using the constants evaluated above, the batmobile computer displays the precise time of chase. However, Batman did not know how to program the computer to solve the set of equations (2.3). He enquired around and was directed to the "experts" and we gave him this chapter! It starts with some mathematical background of linear algebra which is useful in visualization of a matrix equation and determining whether a solution to any given system of equations exists or not, and, if it does, is it a unique solution or would a different set of values of $S_0$, $u$, and $f$, also satisfy the equations? This is followed by the two main

sections of numerical linear algebra, namely, solution of linear systems of equation and estimation of eigenvalues.

### *2.1.1 Properties of Matrices*

A large number of engineering problems involve solution of a system of linear algebraic equations with several independent variables. These equations can be written in the following form:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{2n}x_n = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{3n}x_n = b_3$$

$$\bullet \quad \bullet \quad \bullet \qquad \bullet \quad \bullet$$

$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{mn}x_n = b_m$$

In short, the above system of equation can be written as:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \text{ where}, i = 1,2,3\cdots\cdots m \tag{2.4}$$

Alternatively, the system can be expressed in terms of two vectors (***b*** and ***x***) and a matrix (***A***), as follows:

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{2.5}$$

where, ***b*** is a vector of dimension $m$; ***x*** is a vector of dimension $n$ and ***A*** is a matrix of dimension ($m \times n$). The set of equation is as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ a_{31} & a_{32} & \cdot & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{m1} & a_{m2} & \cdot & \cdot & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ b_m \end{bmatrix} \tag{2.6}$$

Let us now try to visualize what this equation means. The matrix $A$ transforms an $n$-dimensional vector ***x*** to an $m$-dimensional vector ***b***. If $x_i$'s and $b_i$'s are real numbers indicating the position vectors, the matrix ***A*** can be seen as a linear transformation $\Re^n \mapsto \Re^m$. A *linear transformation* ($T$) is a mapping between two vector spaces $X$ and $Y$, $T : X \mapsto Y$, if the following holds:

$$T(\alpha x + \beta y) = \alpha Tx + \beta Ty \quad \forall x, y \in X \text{ and } \forall \alpha, \beta \in \Re^n \tag{2.7}$$

In the case of linear transformation $T : \Re^n \mapsto \Re^m$, let $\{e_1, e_2 \cdots e_n\}$ be the standard basis for $\Re^n$, every vector $x$ in $\Re^n$ can be expressed as a linear combination involving the coordinates (if you are not sure why, please see Box 2.1 for a brief discussion on vector space or consult other books on linear algebra) as:

$$x = x_1 e_1 + x_2 e_2 + x_3 e_3 + \cdots + x_n e_n \tag{2.8}$$

Since $T$ is linear, $Tx = x_1 Te_1 + x_2 Te_2 + x_3 Te_3 + \cdots + x_n Te_n$ (2.9)

Notice that by definition of the linear transformation $T$, each $Te_1$ to $Te_n$ are vectors of dimension $m$ in $\Re^m$. Let us define,

$$u_1 = Te_1, u_2 = Te_2, u_3 = Te_3, \cdots, u_n = Te_n \tag{2.10}$$

Now, construct the $m \times n$ matrix $A$ with $u_i$'s as its columns:

$$A = \begin{bmatrix} u_1 & u_2 & u_3 & \cdot & \cdot & \cdot & \cdot & u_n \end{bmatrix} \tag{2.11}$$

Using the relations of (2.10) and (2.11) in (2.9), we can say, $Tx = Ax$. (2.12)

**Box 2.1**: **Vector Space, Linear Independence and Basis**

1. Let $X$ be a collection of vectors. If for every $x$ and $y$ in $X$, $a$ and $b$ in $\Re$ (set of real numbers), there is a unique vector $ax + by$ that also belongs to $X$ (it is closed under addition and scalar multiplication), then $X$ is called a *real vector space*. If $a$ and $b$ are complex numbers, it is called a *complex vector space*. Examples of real vector space include $\Re^2, \Re^3, \Re^4 \cdots \Re^n$. Any non-empty subset of a vector space that is closed under addition and scalar multiplication is a *subspace* of the vector space. Example: $\Re^2$ is a subspace of $\Re^3$.

3. A vector $x$ in $X$ is a *linear combination* of a set of vectors $\{u_1, u_2 \cdots u_n\}$ in $X$ if it can be expressed as $x = a_1 u_1 + a_2 u_2 + \cdots + a_n u_n$ for some scalars $a_1$, $a_2$ ….$a_n$.

4. A set of vectors $\{u_1, u_2 \cdots u_n\}$ in $X$ are *linearly independent* if $a_1 u_1 + a_2 u_2 + \cdots + a_n u_n = 0$ only when all $a_i$'s are zero. If the summation is zero with some non-zero $a_i$'s, the vectors are linearly dependent.

5. A set of vectors $\{u_1, u_2 \cdots u_n\}$ is said to *span* a vector space $X$ if $X$ contains all linear combinations of the vectors. Furthermore, if the set of vectors are linearly independent and span the vector space, they form a *basis* for the vector space. For example, [1, 0, 0], [0, 1, 0] and [0, 0, 1] form a basis for $\Re^3$. So, is [1, 0, 0], [1, 1, 0] and [1, 1, 1]. But, [1, 1, 0], [0, 1, 0] and [1, 2, 1] do not.

6. Number of vectors in a basis for a given vector space is constant. It is the minimum number of vectors required to span the space and at the same time, maximum number of independent vectors possible. Number of vectors in the basis is termed as the *dimension* of the vector space. We will denote the vectors $e_1$ = [1, 0, …, 0], $e_2$ = [0, 1, …, 0],….$e_n$ = [0, 0, …, 1], as the *standard basis* for $\Re^n$.

In summary, one can say that a matrix is a representation of the linear transform with a particular set of bases. It is important to note that the composition of $A$ is not arbitrary but depends on the basis $\{e_1, e_2 \cdots e_n\}$. For each $x \in X$, the product $Ax$ is a vector in $Y$. So, it

follows that $A(\alpha x + \beta y) = \alpha Ax + \beta Ay$ . If the bases are changed for either of the vector spaces, the matrix $A$ is changed as well and a new matrix represents the same linear transformation. Uniqueness of matrix $A$ for a given set of bases is shown in theorem 2.1.

**Theorem 2.1:** Let $\{u_1, u_2 \cdots u_n\}$ be a basis for vector space $X$ and $\{v_1, v_2 \cdots v_m\}$ be a basis for vector space $Y$. For a linear transformation $T : X \mapsto Y$ with these bases, there exists a unique $m \times n$ matrix $A$, such that, $x = \sum_{i=1}^{n} \xi_i u_i \Rightarrow Tx = \sum_{j=1}^{m} \zeta_j v_j$ with $\zeta = A\xi$.

**Proof:** If the $j^{\text{th}}$ column of matrix $A$ is the coordinate vector of $Tu_j$ with respect to the basis $\{v_1, v_2 \cdots v_m\}$, then it follows:

$$Tu_j = \sum_{i=1}^{m} a_{ij} v_i \quad \text{for} \quad j = 1,2 \cdots n \tag{2.13}$$

where, $a_{ij}$ are the elements of matrix $A$.

Since, $\{v_1, v_2 \cdots v_m\}$ form a basis for the subspace $Y$ and $Tu_j$ is an arbitrary vector in $Y$, the coefficients $a_{ij}$ are unique.

For an arbitrary vector $x$ in $X$, $x = \sum_{j=1}^{n} \xi_j u_j$, we can write,

$$Tx = \sum_{j=1}^{n} \xi_j Tu_j = \sum_{j=1}^{n} \xi_j \sum_{i=1}^{m} a_{ij} v_i = \sum_{i=1}^{m} \zeta_i v_i \tag{2.14}$$

Thus, $\zeta_i = \sum_{j=1}^{n} a_{ij} \xi_j$ for $i = 1,2,\cdots m$. This essentially means, $\zeta = A\xi$.

The matrix A represents the linear transformation $T : X \mapsto Y$ with $\{u_1, u_2 \cdots u_n\}$ as basis for vector space $X$ and $\{v_1, v_2 \cdots v_m\}$ as basis for vector space $Y$. If either of the bases are changed, the matrix changes as well. This illustrated in the following example.

**Example 2.1**: Let us consider a linear transformation $T_1 : \Re^2 \mapsto \Re^3$ with standard bases for $\Re^2$ and $\Re^3$. If the transformation maps the vector [1, 0] to [1, 2, 3] and [0, 1] to [3, 2, 1], the matrix $A_1$ representing the transformation $T_1$ can be written as:

$$A_1 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 3 & 1 \end{bmatrix}$$

Let us change the basis for $\Re^2$ to [1, 0] and [1, 1] but keep the standard basis for $\Re^3$. The same transformation $T_1$ maps [1, 0] to [1, 2, 3] and [1, 1] to [4, 4, 4]. So, the new matrix $B_1$ representing the transformation is:

$$B_1 = \begin{bmatrix} 1 & 4 \\ 2 & 4 \\ 3 & 4 \end{bmatrix}$$

If we formulate a new matrix $C_1$ with the new set of basis for $\Re^2$ as $C_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, note that $B_1 = A_1 C_1$.

Instead, if we changed the basis for $\Re^3$ and kept the standard basis for $\Re^2$, the new matrix representing the same linear transformation could be obtained as $B_1 = D_1^{-1} A_1$ where, $D_1$ is a square matrix of dimension 3 consisting of the new basis vectors as columns. Inverse will always exist since basis vectors by definition are linearly independent. So, in summary, if both bases are changed, the new matrix is given by $B_1 = D_1^{-1} A_1 C_1$. The proof is shown later in section 2.3 for square matrices in the context of similar matrices.

At this point, one can visualize the unknown solution $x$ as the vector that undergoes a linear transformation represented by the matrix $A$. The image after transformation is the vector $b$. Before we embark on solving the system of equation, we will try to understand the circumstances under which a solution will exist. We can re-write the system of equation (2.6) as a linear combination of column vectors:

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \bullet \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ \bullet \\ a_{m2} \end{bmatrix} + x_3 \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \\ \bullet \\ a_{m3} \end{bmatrix} + \cdots\cdots x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ a_{3n} \\ \bullet \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \bullet \\ b_m \end{bmatrix} \tag{2.15}$$

The above form is similar to linear transformation described by (2.9). In this form, one can easily see that the solution of the system is finding suitable coefficients ($x_i$'s) corresponding to the column vectors that result in the right hand side vector $\boldsymbol{b}$. Each column of the matrix $\boldsymbol{A}$ is a vector of dimension $m$. The column vectors would span an $m$-dimensional vector space if they are linearly independent. The vector space spanned by the columns is often referred to as the *Column Space* of $\boldsymbol{A}$. So, a solution of the system will only exist if the right hand side vector $\boldsymbol{b}$ belongs to this vector space spanned by the columns of $\boldsymbol{A}$. If some of the vectors in the columns of $\boldsymbol{A}$ are linear combination of other columns such that there are only $r$ linearly independent columns, the *column space* is of dimension $r$. This $r$ is known as the *rank* of matrix $\boldsymbol{A}$. When, $r = m$, the columns span the complete $m$-dimensional vector space. In this case, each vector $\boldsymbol{b}$ can be represented as a linear combination of the columns and there exists at least one solution.

The other $n - r$ columns can be expressed as a linear combination of the $r$ linearly independent columns and therefore, add nothing new to the *column space*. Certain set of $x_i$ values may lead to the right hand vector to be zero. In other words, they provide the solution to equation $\boldsymbol{Ax} = \boldsymbol{0}$. Collection of all solutions to this equation forms the *Null Space* of $\boldsymbol{A}$. Notice that each vector in the *null space* has $n$ elements and lies in $n$-dimensional vector space. However, since there are not sufficiently many vectors to span the full $n$-dimensional vector space, the *null space* has a dimension less than $n$. Actually, it will have a dimension of $n - r$. Notice that the null space always contains the zero vector $\boldsymbol{x} = \boldsymbol{0}$. If it also contains some

non-zero vectors, from the definition of linear independence of column vectors in eq (2.15), we can conclude that the columns of $\boldsymbol{A}$ are not independent.

For a square matrix $\boldsymbol{A}$ ($m = n$), if the rank of the matrix $r = n$ then we can say that the columns of the matrix are linearly independent. This set then forms a basis for an $n$-dimensional vector space. Since, any vector in that space can be represented as a *unique* linear combination of the bases, the system of equation (2.15) will have a unique solution for any arbitrary right hand side vector $\boldsymbol{b}$. Uniqueness of the solution is shown in theorem2.2. In this case, null space contains only $\boldsymbol{x} = \boldsymbol{0}$. There would also exist another $n \times n$ matrix $B$ such that $BA = I$. That is to say that $B = A^{-1}$ or the matrix $A$ is *invertible*. If the columns are independent, the rows will also be independent in a square matrix. We leave it to the reader to explore why!

**Theorem 2.2**: If $\{u_1, u_2 \cdots u_n\}$ form a basis for vector space $X$, every vector in $X$ has a unique representation as a linear combination of the basis vectors.

**Proof:** Let us choose an arbitrary vector $x$ in $X$ and assume that it has two representations as follows:

$$x = a_1u_1 + a_2u_2 + \cdots + a_nu_n \quad \text{and} \quad x = b_1u_1 + b_2u_2 + \cdots + b_nu_n$$

Since, $x - x = 0$, we can write:

$$x - x = 0 = (a_1 - b_1)u_1 + (a_2 - b_2)u_2 + \cdots + (a_n - b_n)u_n$$

By definition of the basis of a vector space $\{u_1, u_2 \cdots u_n\}$ are a linearly independent set of vectors. Therefore, the above is only possible if all the coefficients are zero, that is, $a_1 = b_1$, $a_2 = b_2 \cdots \; a_n = b_n$.

Concepts of column space and the null space are depicted in the following example.

**Example 2.2**: Let us consider the matrix $A = \begin{bmatrix} 1 & -1 & 0 \\ 0 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$.

It is easy to see that the columns are not independent. If we represent the column vectors as $u_1$, $u_2$ and $u_3$, we observe that $u_1 + u_2 + 2u_3 = 0$. Thus, the columns do not span the complete 3-dimensional vector space or in other words columns do not form the basis for a 3-dimensional vector space. Therefore, solution to equation of the form $Ax = b$ involving this matrix will not exist for any arbitrary right hand side vector $b$ since it will not be possible to express it as a linear combination of the column vectors. For example, the readers can try to solve the system for $b = [1, 1, 1]$ and see for themselves that a solution does not exist. Now, question is, does it have solution for some right hand side vectors $b$ ? If yes, then which ones?

Note that, any two columns of the matrix are linearly independent. Thus, the rank of the matrix is 2, dimension of the null space is 1. Therefore, columns of matrix $A$ span a 2-dimensional vector space (a plane), which is a subspace of the larger 3-dimensional vector space. If we can get some insight into the plane they span, then the solution will exist for any $b$ lying on this plane. In order to construct a vector $b$, we notice that all the column vectors of matrix $A$ can be represented as $[c, c+d, d]$, that is, 2nd element is summation of the first and

the third elements. Therefore, any linear combination of them will also inherit this property and the solution will exist for any $b$ that has this property. The plane is the one perpendicular to the vector [1, -1, 1] and passes through the origin. The system has a solution for $b$ = [1, 2, 1] and the solution is [0, -1, 0]. It is also easy to check that $Ax = 0$ for the vector [1, 1, 2]. So, this vector lies in the null space of the matrix $A$. Let us consider two solution vectors $x$ and $y$ for the same right hand side vector $b$. Since, $Ax = b$ and $Ay = b$, $A(x - y) = 0$. This means that the difference between the two solutions lies in the null space, i.e., it is a vector in the null space. For this example, one solution is [0, -1, 0] and [1, 1, 2] is a vector in the null space. Thus, more solutions can be generated by adding the vector in the null space to one of the solution. This is to say, [1, 0, 2], [2, 1, 4], [3, 2, 6] etc. are also solution to the system of equation with the same right hand side vector.

The solution of the set of equation (2.5) is of course $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ provided the $\boldsymbol{A}^{-1}$ exist. If it exists, one can compute the inverse and multiply with $\boldsymbol{b}$ vector to obtain the solution vector ($\boldsymbol{x}$). Although, the solution of the system of equation (2.5) can be obtained using $\boldsymbol{A}^{-1}$, it is rarely, if ever computed for the engineering problems because more efficient methods of solution are available. Instead, one relies on transforming the set of equation (2.6) into a form that can be solved easily. Various *Numerical Methods* exist for such solutions and form the focus of this chapter.

At this point, we will assume that the reader is familiar with basic matrix algebra such as addition, multiplication and transpose of a matrix. We will also assume familiarity with computation of the trace and determinant of a matrix. These will not be discussed in this book. Unless otherwise mentioned, we will assume real matrices in $\Re^{m\times n}$ and real vectors in $\Re^{n}$. Many theorems discussed here are equally applicable for the complex matrices in $C^{m\times n}$ and complex vectors in $C^{n}$. We will mention them at appropriate places. To refresh some concepts, here we mention some terms related to matrices.

A real square matrix is *symmetric* if $A = A^T$ and *skew symmetric* if $A^T = -A$. For a complex square matrix, it is *hermitian* if $A = A^H$ and *skew symmetric* if $A^H = -A$. $A^T$ and $A^H$ are the *transpose* and *conjugate transpose* of matrix $A$. A square matrix is *normal* if $AA^T = A^TA$ for real $A$ and $AA^H = A^HA$ complex $A$. A matrix is called *orthogonal* if $AA^T = I$ and *unitary* if $AA^H = I$. A real matrix $A$ is *positive definite* if $x^T Ax > 0 \quad \forall x \neq 0$. They are also characterized by positive *eigenvalues* and positive *pivots*. In terms of application, we will use these properties for derivation of some of the numerical methods. However, the readers have to wait a few more sections for clear implication of this statement. While you are already familiar with a *symmetric matrix*, the *pivots* will first appear when we perform Gauss Elimination and we discuss the eigenvalues in the next section.

### *2.1.2 Eigenvalues and Eigenvectors*

For a matrix $A$, if there exist a vector $x$ such that $Ax = \lambda x$, $x$ is called an eigenvector and $\lambda$ is the corresponding eigenvalue. Thus the following follows from the definition,

$$Ax = \lambda x \Rightarrow A^k x = \lambda^k x \tag{2.16}$$

Therefore, eigenvalues are the roots of the polynomial given by $\det(A-\lambda I) = 0$. The resulting polynomial is known as *characteristic polynomial*. Then it automatically follows that

$\det(A) = \prod_{i=1}^{n} \lambda_i$. From properties of the polynomials and coefficient of the term $\lambda^{n-1}$ in characteristic polynomial, one can also write, $trace(A) = \sum_{i=1}^{n} \lambda_i$. (2.17)

Once eigenvalues are determined, the eigenvectors can be obtained by solving the set of equation $(A-\lambda I)x = 0$. Note that $(A-\lambda I)$ is a singular matrix since its determinant is zero and there would not be a unique solution for $x$ (we will get a direction of the eigenvector, its magnitude can be arbitrary).

Before we proceed to further mathematical interpretation, let us spend some time on visualization of the concept of eigenvalue and eigenvector. For this purpose, let us focus on a square matrix ($A$) of size 2 and a vector of size 2 so that the results can be plotted within the realm of the pages of this book.

A vector, when multiplied by a matrix experience rotation and/or stretching or contraction. However, the equation $Ax = \lambda x$ implies that the eigenvectors do not experience rotation. They simply stretch or contract with a ratio of $\lambda$. In order to visualize this, let us consider a unit vector $x$,

$$x = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \quad \theta \in (0,\ 2\pi) \tag{2.18}$$

Next consider a matrix $A = \begin{bmatrix} 3 & 1 \\ 2 & 2 \end{bmatrix}$ (2.19)

If we plot the position vector $x$ for all values of $\theta$, it traces a circle of unit radius (Figure 2.1). In the same plot, the position vector of $Ax$ is shown as the ellipse. The circle is transformed to an ellipse due to rotation, elongation and contraction of vectors $x$ as a result of multiplication by the matrix $A$.

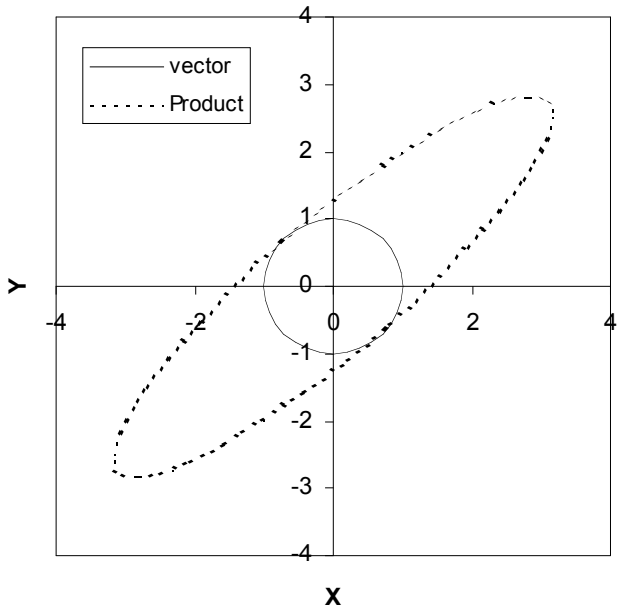



**Figure 2.1**: Traces of vector $x$ and $Ax$ of equations (2.18) and (2.19).

By equating the determinant of the matrix ($A$-$\lambda I$) to zero, one obtains the eigenvalues as 4 and 1. The corresponding eigenvectors are given by (1,1) and (-0.5,1), respectively. By normalizing the eigenvectors in order to correspond to the circle of $x$ one obtains, $\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right)$ and $\left(-\frac{1}{\sqrt{5}},\frac{2}{\sqrt{5}}\right)$. The reader can easily check that these two vector directions do not change as a result of multiplication by matrix $A$. The eigenvector $\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right)$ corresponding to eigenvalue $\lambda$=4 elongates by a factor of 4. The eigenvector $\left(-\frac{1}{\sqrt{5}},\frac{2}{\sqrt{5}}\right)$ corresponding to eigenvalue $\lambda$=1 remains unaltered. At this point, the readers may also note that if $x$ is an eigenvector, $-x$ will also be one (in fact any multiple of $x$ would be an eigenvector).

Let us look at an enlarged portion of the 2nd quadrant (Figure 2.2). Vector marked as 1 is the eigenvector corresponding to $\lambda$=1. Vector 2 becomes Vector 3 as a result of multiplication by matrix $A$ although no elongation or contraction takes place (both lie on the circle). So, one can safely say that all vectors other than the eigenvectors will rotate as a result of the multiplication by a matrix. They may or may not experience elongation or contraction. The eigenvectors on the other hand will not rotate by definition. They will experience elongation or contraction in the same proportion as the corresponding eigenvalue. Mathematically speaking, the vectors $x$ and $Ax$ will belong to the same subspace which is *invariant* for matrix $A$. A subspace $X \subseteq C^n$ is *invariant* for matrix $A$ iff $x \in X$ implies $Ax \in X$.

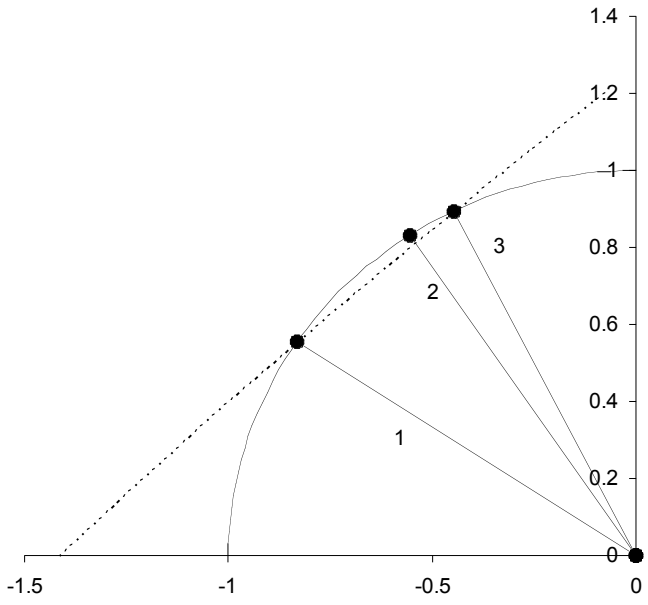



**Figure 2.2**: Closer look at the second quadrant of Figure 2.1.

There may be more than one linearly independent eigenvectors corresponding to an eigenvalue. For a simple example, let us consider a diagonal matrix,

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{2.20}$$

The loci of $x$ and $Ax$ are now concentric circles (Fig 2.3). It is easy to see that both the eigenvalues are equal to 2. Every vector $x$ is an eigenvector. There are infinitely many eigenvectors.

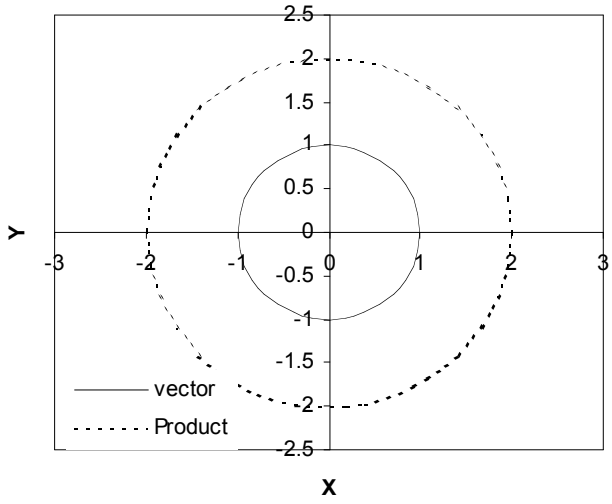



**Figure 2.3**: Traces of vector $x$ and $Ax$ of equations (2.18) and (2.20).

If the diagonal matrix is changed to

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.21}$$

once again we get back two unique eigenvectors corresponding to eigenvalues 2 and 1 lying along the $x$ and $y$ axes, respectively (Figure 2.4).

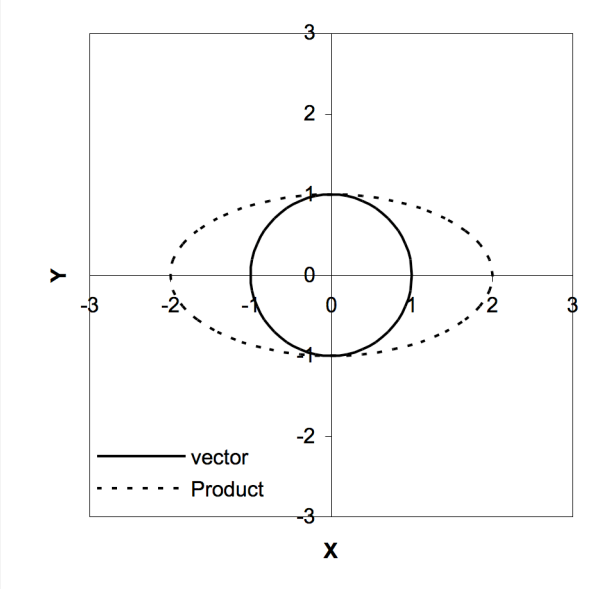



**Figure 2.4**: Traces of vector $x$ and $Ax$ of equations (2.18) and (2.21).

We have seen in the previous examples that an eigenvalue can occur more than once in a matrix. If an eigenvalue occurs only once in a matrix, it is known as a *simple eigenvalue*. If an eigenvalue occurs $n$ times in a matrix, it is said to have an *algebraic multiplicity* of $n$. For example, the eigenvalue $\lambda = 2$ in the matrix in equation (2.20), has an algebraic multiplicity of 2. An eigenvalue with algebraic multiplicity > 1 may or may not have multiple linearly independent eigenvectors associated with them. The matrix in example (2.20) has infinitely many linearly independent eigenvectors for the eigenvalue with algebraic multiplicity of 2. Now consider the following matrix:

$$A = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} \tag{2.22}$$

It is easy to verify that for this matrix, the eigenvalue $\lambda = 2$ has an algebraic multiplicity of 2. However, there exists only one linearly independent eigenvector associated with it which is (1,-1). The number of linearly independent eigenvectors associated with an eigenvalue is known as its *geometric multiplicity*. If the *algebraic multiplicity* of an eigenvalue is greater than its *geometric multiplicity*, it is known as a *defective* eigenvalue. The matrix with a defective eigenvalue is a *defective* matrix. Otherwise, the matrix is *non-defective*. This classification has important implications when the eigenvalues are estimated using diagonalization (section 2.3).

In this section, we have observed a few interesting properties of matrices. We saw that operation of a matrix on a vector results in another vector which may or may not have the same magnitude and/or direction. From a computational point of view, we would be interested in the condition of a matrix (see discussion of condition number in chapter 1) in terms of whether it will significantly amplify small round-off errors or not. For this purpose, we should define some measure of the size of a matrix. Similar to vector norms, described in chapter 1, we now define matrix norms.

### *2.1.3 Matrix Norms*

A matrix norm is useful to have a sense of the "*size*" of a given matrix. This "*size*" is not related to the number of rows or columns or elements in a matrix but represents the magnitude of the effect it creates when operated on a vector. There are many ways to define such a norm for a matrix. Before we get on with the definitions of various matrix norms, let us first list a set of properties that any defined norm must satisfy and have a visual sense of what matrix norm might mean. Following on the basic discussion of vector norm (in Chapter 1), one may list the following properties a matrix norm must satisfy:

- $\|A\| > 0 \quad \forall A \neq 0$ and $\|A\| = 0$ *iff* $A = 0$. (2.23)
- $\|\alpha A\| = |\alpha| \|A\|$ for all scalar $\alpha$. (2.24)
- $\|A + B\| \leq \|A\| + \|B\|$ (2.25)
- $\|AB\| \leq \|A\| \|B\|$ (2.26)
- $\|Ax\| \leq \|A\| \|x\|$ for the matrix norm to be consistent with the norm of vector $x$ (2.27)

Let us take a re-look at the Figures 2.1 for a sense of this "*size*". All the unit vectors contained in the circle remain within the closed curve traced by $Ax$. It may be an ellipse (Figures 2.1) or a circle (Figure 2.3) depending on the eigenvalues. Thus, if we can find the maximum magnitude of extension a unit vector experiences as a result of the multiplication, expansion of all other vectors can be easily obtained by multiplying with the vector norm. If the magnitude of maximum expansion is defined as the "*size*" or norm of the matrix, its effect is to stretch any vector by the same factor or less. The only job then is to find this maximum extension. Intuitively, one may think that it is in the direction of the major axis of ellipse but it is not always so. For example, in Figure 3.1, the vector along the major axis of the ellipse, which is also an eigenvector, stretches by a factor of 4, same as the corresponding eigenvalue. It will be easy for the reader to check that a larger extension occurs for the vector (0.86491, 0.50193), which is transformed to (3.0967, 2.7337) and experiences an elongation to the extent of 4.131. This vector not being an eigenvector also experiences a rotation. The task is then to find the magnitude of this maximum extension in a logical manner. For an arbitrary $m \times n$ matrix $A$, it is easy to verify that $A^TA$ is a square and symmetric matrix. The square root of the maximum eigenvalue of $A^TA$ provides this magnitude of maximum extension for $A$. A brief outline of the derivation is shown below:

Let us consider a matrix $A$ that transforms the unit vector $x$ ($x^Tx = 1$) to a vector $b$. We are interested in finding the direction $x$ for which norm of vector $b$ is maximum. Amount of maximum extension using Euclidean norm for vectors can then be calculated as $\|b\| = \sqrt{b^Tb}$. This is a maximization problem which can be stated as, maximize $b^Tb$ subject to $x^Tx = 1$. Using Lagrange multiplier, one may write the following:

$$\nabla\left[b^Tb - \lambda\left(x^Tx - 1\right)\right] = 0 \tag{2.28}$$

Since, $Ax = b$,

$$\nabla\left[x^TA^TAx - \lambda\left(x^Tx - 1\right)\right] = 0 \;\text{ or }\; A^TAx = \lambda x \tag{2.29}$$

Therefore, the maximum extension is obtained when the multiplier $\lambda$ is an eigenvalue of the matrix $A^TA$ and the magnitude of this extension is $\|b\| = \sqrt{b^Tb} = \sqrt{x^TA^TAx} = \sqrt{x^T\lambda x} = \sqrt{\lambda}$. This quantity is known as the *spectral norm* of matrix $A$. Spectral norm is typically denoted as $\|A\|_2$ and formally defined as:

$$\|A\|_2 = \sqrt{\max \lambda\left(A^TA\right)} \tag{2.30}$$

**Example 2.3**: Find the spectral norm of the matrix in equation (2.19).

**Solution:** $A = \begin{bmatrix} 3 & 1 \\ 2 & 2 \end{bmatrix}$, $A^T = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$, $A^TA = \begin{bmatrix} 13 & 7 \\ 7 & 5 \end{bmatrix}$ (2.31)

The largest eigenvalue of $A^TA$ is 17.06226 and its square root is 4.13065, which is the maximum extension for any vector due to multiplication by $A$.

Similar to *p-norm* for a vector (section 1.4), a matrix norm can also be defined for a $m \times n$ matrix as follows:

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} \tag{2.32}$$

Numerator is the norm of the transformed vector and the denominator is the original vector. The above norm gives a measure of the maximum extension ratio attainable in the transformation. Using this definition, the following two norms can be easily computed:

In (2.32), if one is interested to see how much a unit vector would expand due to linear transformation given by matrix $A$, one can use the standard basis (Box 2.1) for a vector space as unit vectors. In this case, if one uses $L_1$ norm (Chapter 1) for computing the norm for the vectors, one obtains the *column-sum norm* as follows:

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}| \tag{2.33}$$

Alternatively, one can consider extension for a vector in which all the components are unity. For this vector, the $L_\infty$ norm (Chapter 1) is unity. In this case, the $L_\infty$ norm of the resulting vector gives the *row-sum norm* as follows:

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}| \tag{2.34}$$

*Frobenius* norm for a matrix is defined as follows:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} = \sqrt{trace\left(A^T A\right)} \tag{2.35}$$

For a matrix $A$ with $p$ distinct eigenvalue, the largest absolute eigenvalue is known as the *spectral radius* and is typically denoted as $\rho(A)$. That is,

$$\rho(A) = \max_{1 \le i \le p} |\lambda_i| \tag{2.36}$$

The *spectral radius* provides a lower bound for the matrix norms. In order to see this, let us consider a eigenvalue-eigenvector pair,

$Ax_j = \lambda_j x_j$, so, $\|Ax_j\| = |\lambda_j| \|x_j\|$; but, $\|Ax_j\| \le \|A\| \|x_j\|$. This essentially means:

$$|\lambda_j| \le \|A\| \quad \forall j \quad \text{and therefore,} \quad \rho(A) = \max_{1 \le j \le n} |\lambda_j| \le \|A\| \tag{2.37}$$

We will end this section by stating a relationship between the matrix norms and the spectral radius. For any norm of matrix $A$,

$$\rho(A) = \lim_{p \to \infty} \|A^p\|^{1/p} \tag{2.38}$$

Thus, spectral radius can be interpreted as the asymptotic growth rate of the norm of the power sequence of matrix $A$. The above relation is known as the *Gelfand's formula* [see *Gelfand (1940)* for proof].

With the background described above, we are now ready to proceed to numerical linear algebra. In the following sections, we will cover two topics, methods for solution of linear systems of equations and estimation of eigenvalues of a matrix.

## Exercise 2.1

1. If $\{x_1, x_2 \cdots x_n\}$ forms a basis for $n$-dimensional vector space $X$, show that any vector $x$ in $X$ has a unique representation as a linear combination of the basis vectors.

2. (a) Show that all the eigenvalues of a positive-definite matrix are positive.

   (b) Show that the spectral radius of a square matrix cannot exceed the largest sum of the absolute values of the elements of the matrix along any row.

3. Let $\boldsymbol{A}$ be a given nonsingular $n \times n$ matrix, and $\boldsymbol{X_0}$ an arbitrary $n \times n$ matrix. We define a sequence of matrices by

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \boldsymbol{X}_k(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{X}_k), \quad k = 0, 1, 2, \ldots.$$

Prove that $\lim_{k \to \infty} X_k = A^{-1}$ if and only if $\boldsymbol{\rho}(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{X_0}) < 1$.

4. Derive equations (2.33) and (2.34) from (2.32).

5. For a symmetric matrix, establish a relation between *spertral norm* and *spectral radius* ?

## 2.2 Solution of Linear Systems

When the solution of a system exists, one is often interested in computing it in the most economical fashion. Economy in this context refers to minimum number of floating point computation. Computation of inverse of the coefficient matrix $\boldsymbol{A}$ requires large number of computations and is best avoided for solving a system of equation. We will see the computational requirements later in the chapter. So, goal of all the *Numerical Methods* in this direction is to minimize the number of computations to solve a system. The existing methods can broadly be classified into two distinct categories. These are as follows:

*Direct Methods:* This group of methods obtains the solution of a system in a finite number of computations or steps. Final errors in the solution are mainly due to round-off error. In this group, we will outline the *Gauss Elimination*, *Gauss Jordon* and *LU Decomposition* for general coefficient matrices and *Thomas Algorithm* for a special group of matrix.

*Indirect Methods:* In these methods, an approximate solution is assumed and iteratively improved to obtain desired accuracy. Since, the solution vector in this method is approached in a sequential manner, the final error can be arbitrary. However, the minimum error or the lower bound of the error is the accumulation through round-off error, which in turn, is a function of the number of iteration. We will outline three methods in this group: *Jacobi Iteration*, *Gauss Seidel Iteration* and *Successive Over-Relaxation (SOR)*.

### 2.2.1 Direct Methods

We will consider equations of the form (2.5) with square coefficient matrix (***A***) of dimension ($n \times n$) with rank $n$, such that unique solutions exist. If the coefficient matrix has only non-zero elements along the major diagonal (***A***) and all other off-diagonal elements are zero, *i.e.*, ***A*** is a *diagonal matrix*, the solution vector $x_i$'s can be obtained directly by dividing the elements of ***b*** vectors by the diagonal element in the same row.

If we have a coefficient matrix which is an upper triangular matrix, *i.e.*, all the entries below the main diagonal are zero, the set of equation can be represented as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ 0 & a_{22} & \cdot & \cdot & a_{2n} \\ 0 & 0 & a_{33} & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdots a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ b_n \end{bmatrix} \tag{2.39}$$

The above set of equations can be solved directly starting from the last equation. Solution of the last equation or $n^{th}$ equation is:

$$x_n = \frac{b_n}{a_{nn}} \tag{2.40}$$

Using the value of $x_n$, one can now obtain the value of $x_{n-1}$ from the $(n\text{-}1)^{th}$ equation,

$$x_{n-1} = \frac{b_{n-1} - a_{n-1n} x_n}{a_{n-1n-1}} \tag{2.41}$$

Knowing the values of $x_n$ and $x_{n-1}$, one can now obtain $x_{n-2}$ from the $(n\text{-}2)^{th}$ equation. This way one can continue up to the first equation and obtain all the $x_i$'s provided the diagonal elements or $a_{ii}$'s are non-zero for all $i$ values. This method of solving an upper triangular system is known as *Back Substitution*. The algorithm for the *Back Substitution* can be written as follows:

$$x_n = b_n / a_{nn} \text{ and } x_i = \frac{\left( b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right)}{a_{ii}}; \; i = (n-1), \cdots\cdots 3,2,1 \tag{2.42}$$

If the coefficient matrix is a lower triangular matrix, *i.e.*, the entries below the main diagonal are zero, the system of equation can be written as follows:

$$\begin{bmatrix} a_{11} & 0 & \cdot & \cdot & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdot & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdot & \cdot & 0 \\ & \cdot & \cdot & \cdot & \cdot & \\ a_{n1} & a_{n2} & a_{n3} & \cdot & a_{nn} & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ b_n \end{bmatrix} \tag{2.43}$$

This lower triangular system can be solved similarly starting from the first equation and sequentially substituting the known values to the subsequent equations. In this way, one can obtain all the $x_i$'s progressing from the first equation down to the last equation provided all the diagonal elements are non-zero. This method of solving a lower triangular system is called the *Forward Substitution*. The algorithm can be written as follows:

$$x_1 = b_1 / a_{11} \text{ and } x_i = \frac{\left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j \right)}{a_{ii}}; \; i = 2,34 \cdots\cdots n \tag{2.44}$$

All the direct methods rely on transforming the coefficient matrix ($A$) into an upper triangular or a lower triangular or a diagonal matrix or a combination of these with non-zero diagonal elements through a finite number of operations.

Number of floating point operations required to solve a triangular system can be calculated using (2.42) or (2.44). In (2.44), evaluation of $x_1$ requires only one division. Evaluation of $x_2$ requires one subtraction, one multiplication and one division. Proceeding similarly, one can easily see that to obtain $x_i$ one requires to perform ($i$-1) addition, ($i$-1) multiplication, one subtraction and one division, *i.e.,* total $2i$ operations. Thus, in order to solve the complete system one needs to perform $n(n+1)$ operations. For large $n$, one can say that the number of operations required is approximately of the order $n^2$.

### 2.2.1.1 Gauss Elimination

This is one of the most popular methods for solving a linear system of equation of the form (2.6). In this section, we will assume that the coefficient matrix has all sorts of nice properties, such as, square ($n \times n$) with rank $n$ such that we do not have to worry about the existence of the solution. In this method, the coefficient matrix will be reduced to an upper triangular matrix through a series of row operations. Our starting matrix ($\boldsymbol{A}$) is the one shown in eq (2.6). Although, we will mainly operate on the rows, the elements below the leading diagonal will be reduced to zero column-wise. As we proceed to make sub-diagonal elements of each column to zero, we will count these as *Steps*.

*Step 1*: The goal is to reduce the elements of 1st column to zero except the element $a_{11}$. In order to do that, we will keep the 1st row unaltered and define a multiplying factor for each row. For example, to make $a_{21}$ zero, we can multiply the 1st row by ($a_{21}/a_{11}$) and deduct from the 2nd row. So, ($a_{21}/a_{11}$) is the multiplying factor for the 2nd row. Similarly, multiplying 1st row by the multiplying factor for the 3rd row ($a_{31}/a_{11}$) and deducting it from 3rd row will make $a_{31}$ zero. Proceeding this way, we can reduce $a_{i1}$ to zero by multiplying the 1st row by the multiplying factor for the $i^{th}$ row ($a_{i1}/a_{11}$) and deducting it from the $i^{th}$ row. This sequence of operation is equivalent to multiplying the first equation by a constant and deducting the

resulting equation from all subsequent equations in order to make the coefficient of the variable $x_1$ to be zero, i.e. eliminate the variable. So, similar operations also need to be conducted for the right hand vector $\boldsymbol{b}$. These operations on $\boldsymbol{b}$ can be carried out independently or alternatively, the vector can be augmented as the ($n$+1) column of the matrix $\boldsymbol{A}$.

We will denote the multiplying factor of $i^{th}$ row as $l_{i1} = (a_{i1}/a_{11})$. Note that the second index on the multiplying factor as well as '1' in the indices on '$a$' correspond to the step number. The algorithm for *Step 1* can be written as follows:

$$l_{i1} = \frac{a_{i1}}{a_{11}} \;,\; a_{ij} = a_{ij} - l_{i1}a_{1j} \text{ and } b_i = b_i - l_{i1}b_1 \text{ for } i,j = 2 \text{ to } n \tag{2.45}$$

If the vector $\boldsymbol{b}$ is augmented as the ($n$+1) column of the matrix $\boldsymbol{A}$

$$l_{i1} = \frac{a_{i1}}{a_{11}} \;,\; a_{ij} = a_{ij} - l_{i1}a_{1j} \text{ for } i = 2 \text{ to } n \text{ and } j = 2 \text{ to } (n+1) \tag{2.46}$$

We notice that augmenting the vector $b$ essentially results in the same operation to be computed with range of $j$ extended to ($n$+1) in place of $n$. This point onwards, the algorithme for the augmented matrix will not be shown explicitly.

It is easy to see that the entire operation will fail if the element $a_{11}$ is zero or very close to zero, since it appears in the denominator of the multiplying factor. It is known as the "pivot" of *Step 1*. Also note that while a multiplying factor is defined for each row, it is always multiplied to the first row. At the end of the *Step 1*, only first row remained unchanged. All other elements in the matrix in rows 2 to $n$ are the modified values. Since, the elements of the first column in these rows were designed to be zero, only elements in columns 2 to $n$ need to be recomputed. So, both row index ($i$) as well as the column index ($j$) starts from '2'.

After the first step, the matrix $\boldsymbol{A}$ is of the following form:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ 0 & a_{22} & \cdot & \cdot & a_{2n} \\ 0 & a_{32} & \cdot & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & a_{n2} & \cdot & \cdot & a_{nn} \end{bmatrix} \tag{2.47}$$

*Step 2*: In this step, we will reduce sub-diagonal elements of column 2 ($a_{32}$, $a_{42}$ ….. $a_{n2}$) to zero. For the third row, we define a multiplying factor $l_{32} = (a_{32}/a_{22})$, multiply the 2nd row with it and deduct from the third row. Similarly, for the $i^{th}$ row, we define the multiplying factor as $l_{i2} = (a_{i2}/a_{22})$, multiply the 2nd row with it and deduct from the third row. Once again note that the second index on the multiplying factor as well as '2' in the indices correspond to the step numbers. The computations are done on rows 3 to $n$ and in each row, on columns 3 to $n$. So, the algorithm for *Step 2* can be written as,

$$l_{i2} = \frac{a_{i2}}{a_{22}} \;,\; a_{ij} = a_{ij} - l_{i2}a_{2j} \text{ and } b_i = b_i - l_{i2}b_2 \text{ for } i,j = 3 \text{ to } n \tag{2.48}$$

At the end of this step, the matrix is as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ 0 & a_{22} & \cdot & \cdot & a_{2n} \\ 0 & 0 & \cdot & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & a_{nn} \end{bmatrix} \tag{2.49}$$

Continuing in this fashion, one can write the computations of the $k^{th}$ step as follows:

$$l_{ik} = \frac{a_{ik}}{a_{kk}}, \; a_{ij} = a_{ij} - l_{ik} a_{kj} \text{ and } b_i = b_i - l_{ik} b_k \text{ for } i,j = (k+1) \text{ to } n \tag{2.50}$$

After completion of *Step k*, the coefficient matrix is of the following form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1k} & a_{1k+1} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & a_{24} & \cdots & a_{2k} & a_{2k+1} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & a_{34} & \cdots & a_{3k} & a_{3k+1} & \cdots & a_{3n} \\ 0 & 0 & 0 & a_{44} & \cdots & a_{4k} & a_{4k+1} & \cdots & a_{4n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & a_{kk} & a_{kk+1} & \cdots & a_{kn} \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{k+1k+1} & \cdots & a_{k+1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{nk+1} & \cdots & a_{nn} \end{bmatrix} \tag{2.51}$$

To obtain an upper triangular matrix, we have to reduce the sub-diagonal elements of first ($n$-1) columns to zero. Since, in the $k^{th}$ step, the sub-diagonal elements of column $k$ are reduced to zero, we only need to proceed for ($n$-1) steps. So, the equation (2.50) is the algorithm for Gauss elimination with $k$ = 1 to ($n$-1). The full algorithm becomes,

$$l_{ik} = \frac{a_{ik}}{a_{kk}}, \; a_{ij} = a_{ij} - l_{ik} a_{kj} \text{ and } b_i = b_i - l_{ik} b_k \text{ for } k = 1 \text{ to } (n\text{-}1), \; i,j = (k+1) \text{ to } n \tag{2.52}$$

If the vector $\boldsymbol{b}$ is augmented as the ($n$+1) column of the matrix $\boldsymbol{A}$, the operations could be extended as in (2.46).

The above is known as *Forward Elimination* of the Gauss elimination procedure. At this point, one can easily obtain the solution of the system of equation using *Back Substitution*. So, the combination of *Forward Elimination* and *Back Substitution* provides the required solution of the set of equations and is known as Gauss Elimination Procedure.

As mentioned earlier, the process will fail if any of the elements $a_{kk}$ for $k$ = 1 to ($n$-1) become zero. These elements are called *Pivots* and at each step $k$, the $k^{th}$ row and column are known as *pivotal row* and *pivotal column*, respectively. We will discuss the situations when pivot's are very small or zero, in a latter section.

Using (2.52), one can calculate the number of computations required to reduce a full system of equation to a triangular system using Gauss elimination. At each $k$, we require ($n$-$k$)

computations for $l_{ik}$'s and $2(n\text{-}k)(n\text{-}k+1)$ for $a_{ij}$'s. Thus, total number of computation is $\sum_{k=1}^{n-1}\left[(n-k)+2(n-k)(n-k+1)\right]$ or $\left(\frac{2n^3}{3}+\frac{n^2}{2}-\frac{7n}{6}\right)$. In order to solve the system completely, one needs to add the number of computations required to solve a triangular system which was calculated as $n(n+1)$. So, overall computation requirement is $\left(\frac{2n^3}{3}+\frac{3n^2}{2}-\frac{n}{6}\right)$. For large $n$, it is safe to say that Gauss Elimination require approximately $\frac{2n^3}{3}$ operations.

**Example 2.4:** Solve Batman's problem given by equation (2.2) using Gauss Elimination.

**Solution:** The augmented matrix is given by:

$$\begin{bmatrix} 1 & 6.2 & 19.22 & 439 \\ 1 & 14.2 & 100.82 & 640 \\ 1 & 22.4 & 250.88 & 875 \end{bmatrix}$$

$k$ =1, pivot $a_{11}$ = 1, define the multiplying factors as $l_{21}$ = 1/1 = 1 and $l_{31}$ = 1/1 = 1. Using these multiplying factors and performing the row operations by equation (2.52), we obtain:

$$\begin{bmatrix} 1 & 6.2 & 19.22 & 439 \\ 0 & 8 & 81.6 & 201 \\ 0 & 16.2 & 231.66 & 436 \end{bmatrix}$$

$k$ = 2, pivot $a_{22}$ = 8, define the multiplying factor $l_{32}$ = 16.2/8=2.025. Once again performing the row operations:

$$\begin{bmatrix} 1 & 6.2 & 19.22 & 439 \\ 0 & 8 & 81.6 & 201 \\ 0 & 0 & 66.42 & 28.975 \end{bmatrix}$$

Now, applying the backward elimination algorithm of equation (2.42), we obtain the solutions as $f$ = 28.975/66.42 = 0.436239, $u$ =(201 – 81.6 x 0.436239)/8 = 20.67536, and $S_0$ = (439 – 19.22 x 0.436239 – 6.2 x 20.67536)/1 = 302.42825.

### 2.2.1.2 Gauss Jordon Elimination

In this method, the matrix $A$ is reduced to an identity matrix through a series of operations. This is easily accomplished by minor modification of the Gauss elimination algorithm described in the previous section. The modifications are as follows:

- At each step, first the pivotal element is made unity by dividing the pivotal row with the pivotal element. So, at *Step k*, the $k^{th}$ row is divided by $a_{kk}$. As a result, the multiplication factor for $i^{th}$ row at *Step k* becomes $l_{ik} = a_{ik}$.

- In addition to the sub-diagonal elements, the above diagonal elements are also made zero. So, at *Step k*, the row operations are conducted for all rows except the pivotal row. At this step, the columns 1 to $k$-1 have already been operated to resemble part of the identity matrix. So, the operations are done on columns $k$ to $n$.

As a result, the algorithm becomes:

$$a_{kj} = \frac{a_{kj}}{a_{kk}}, b_k = \frac{b_k}{a_{kk}}, \; a_{ij} = a_{ij} - a_{ik}a_{kj} \text{ and } b_i = b_i - a_{ik}b_k \tag{2.53}$$

for $k = 1$ to $n$, $i = 1$ to $n$ but $\neq k$, and $j = k$ to $n$.

If the vector $\boldsymbol{b}$ is augmented as the $(n+1)$ column of the matrix $\boldsymbol{A}$

$$a_{kj} = \frac{a_{kj}}{a_{kk}} \text{ and } a_{ij} = a_{ij} - a_{ik}a_{kj} \text{ for } k = 1 \text{ to } n,\ i = 1 \text{ to } n \text{ but } \neq k, j = k \text{ to } (n+1). \tag{2.54}$$

It is easy to see that the Gauss Jordon elimination process can be used to obtain inverse of a matrix. An identity matrix of size $(n \times n)$ can be augmented to the existing $(n \times n)$ matrix $\boldsymbol{A}$ such that the resultant matrix becomes $(n \times 2n)$, as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} & \cdots & 1 & 0 & 0 & \cdot & 0 \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} & \cdots & 0 & 1 & 0 & \cdot & 0 \\ a_{31} & a_{32} & \cdot & \cdot & a_{3n} & \cdots & 0 & 0 & 1 & \cdot & 0 \\ & \cdot & \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} & \cdots & 0 & 0 & 0 & \cdot & 1 \end{bmatrix} \tag{2.55}$$

Now one can perform the Gauss Jordon elimination on the augmented matrix such that the original $\boldsymbol{A}$ matrix becomes an identity matrix. The original identity matrix will transform in the process to give $\boldsymbol{A}^{-1}$. The algorithm can be written as follows:

$$a_{kj} = \frac{a_{kj}}{a_{kk}} \text{ and } a_{ij} = a_{ij} - a_{ik}a_{kj} \text{ for } k = 1 \text{ to } n,\ i = 1 \text{ to } n \text{ but } \neq k, \text{ and } j = k \text{ to } 2n. \tag{2.56}$$

We leave it up to the readers to calculate the number of computations required to invert a matrix using Gauss-Jordon Method.

**Example 2.5:** Solve Batman's problem given by equation (2.2) using Gauss Jordon Elimination.

**Solution:** We will use the algorithm of augmented matrix given by equation (2.54). The augmented matrix is given by:

$$\begin{bmatrix} 1 & 6.2 & 19.22 & 439 \\ 1 & 14.2 & 100.82 & 640 \\ 1 & 22.4 & 250.88 & 875 \end{bmatrix}$$

Two steps of the Gauss Jordon iterations are shown in the matrix form below:

Step 1: $\begin{bmatrix} 1 & 6.2 & 19.22 & 439 \\ 0 & 8 & 81.6 & 201 \\ 0 & 16.2 & 231.66 & 436 \end{bmatrix}$

Step 2: $\begin{bmatrix} 1 & 0 & -44.02 & 283.225 \\ 0 & 1 & 10.2 & 25.125 \\ 0 & 0 & 66.42 & 28.975 \end{bmatrix}$

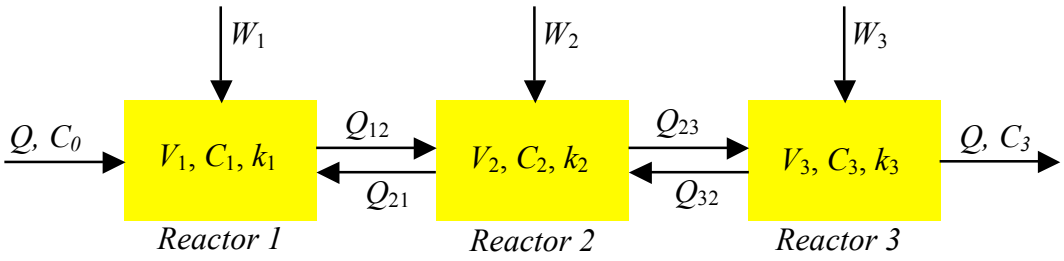
Step 3: $\begin{bmatrix} 1 & 0 & 0 & 302.4282 \\ 0 & 1 & 0 & 20.6754 \\ 0 & 0 & 1 & 0.4362 \end{bmatrix}$

Note that the augmented right hand side column gives the solution vector.

### 2.2.1.3 LU Decomposition

In many engineering problems, one requires to solve a system repeatedly where coefficient matrix ($\boldsymbol{A}$) remains the same but the right hand side vector ($\boldsymbol{b}$) changes. Let's say the solutions are required for right hand side vectors $\boldsymbol{b_1}$, $\boldsymbol{b_2}$, $\boldsymbol{b_3}$, ….. $\boldsymbol{b_m}$. If these vectors are mutually independent, one can augment all the columns with the coefficient matrix and obtain the solutions together. However, most often, vector $\boldsymbol{b_2}$ will depend upon the solution vector obtained from $\boldsymbol{b_1}$ and so on. In this case, $\boldsymbol{b_2}$ is not known until $\boldsymbol{Ax} = \boldsymbol{b_1}$ is solved. So, one needs to run the Gauss elimination repeatedly even though the coefficient matrix remained same. We will illustrate it through an example:

**Example 2.6**: Let us consider a series of adjacent reactors. Each reactor is completely mixed such that the concentration within each reactor is uniform. However there can be exchange of flow in both directions between two adjacent reactors. Such a system is shown in figure below.




In the figure, $Q_i$'s are the flow rates, $C_i$'s are concentrations, $V_i$'s are reactor volumes and $k_i$'s are the first order reaction rates and $W_i$'s are the external loads in the reactors. All the $Q_i$'s, $V_i$'s, $W_i$'s, $k_i$'s and $C_0$ are known. Formulate the set of equations to calculate $C_1$, $C_2$ and $C_3$ at steady state.

**Solution**: The mass balance reactions for the above reactors can be written as:

*Reactor 1*: $V_1 \dfrac{dC_1}{dt} = QC_0 + W_1 + Q_{21}C_2 - Q_{12}C_1 - k_1V_1C_1$

$$\textit{Reactor 2: } V_2\frac{dC_2}{dt} = Q_{12}C_1 + W_2 + Q_{32}C_3 - Q_{21}C_2 - Q_{23}C_2 - k_2V_2C_2$$

$$\textit{Reactor 3: } V_3\frac{dC_3}{dt} = Q_{23}C_2 + W_3 - Q_{32}C_3 - k_3V_3C_3 - QC_3$$

At steady state, the left hand side of all the equations is zero. Thus, the set of equations to solve is,

$$\begin{aligned}
(-Q_{12} - k_1V_1)C_1 + Q_{21}C_2 &= (-QC_0 - W_1)\\
Q_{12}C_1 + (-Q_{21} - Q_{23} - k_2V_2)C_2 + Q_{32}C_3 &= -W_2\\
Q_{32}C_2 + (-Q_{32} - k_2V - Q)C_3 &= -W_3
\end{aligned}$$

This set of equations can be solved to obtain the $C_i$'s. However, the concentrations in each reactor as well as the final concentration $C_3$ are often governed by guideline values. The coefficient matrix will remain unaltered as long as the reaction rates, volumes and the flow rates remain constant. The $W_i$'s are often adjusted depending on the values of $C_i$'s so that the required concentrations are obtained. In addition, the initial concentration $C_0$ may also have diurnal and seasonal variations. The change the right hand side vectors, one as a function of $C_i$'s and the other as a function of time. Under these circumstances, it is very handy to have a $LU$ decomposition of the coefficient matrix so that the solution can be obtained quickly for any right hand side vector.

In the above example, the coefficient matrix is tri-diagonal. Notice, that the Reactor 3 has exchange flow only with Reactor 2. If we place three reactors as vertices of a triangle and add exchange flow to the third reactor from both 1 and 2, the matrix will be full. We leave this for the readers to work out. This example has high practical significance since, lakes, rivers and estuaries are often simulated by dividing into small reactors with exchange flows. Division is made in horizontal segments as well as in vertical segments.

In summary, the method of $LU$ decomposition provides an alternative by which one can avoid repeated operation on the same coefficient matrix ($\boldsymbol{A}$). For large matrices, this saves significant amount of computation time. Any square matrix, on which Gauss elimination will work, can be expressed as a product of a lower triangular matrix ($\boldsymbol{L}$) and an upper triangular matrix ($\boldsymbol{U}$). More specifically, the condition for such a decomposition is given by theorem 2.3 also known as the *LU Theorem*.

**Theorem 2.3:** Let $\boldsymbol{A_k}$ be a sequence of matrix formed by first $k$ rows and $k$ columns of a $n \times n$ square matrix $\boldsymbol{A}$. If det ($\boldsymbol{A_k}$) $\neq$ 0 for $k$ = 1, 2, … ($n$-1), then there exist an $n \times n$ upper triangular matrix $\boldsymbol{U}$ and a $n \times n$ lower triangular matrix $\boldsymbol{L}$ such that, $\boldsymbol{A} = \boldsymbol{LU}$. Furthermore, if the diagonal elements of either $\boldsymbol{L}$ or $\boldsymbol{U}$ are unity, i.e. $l_{ii}$ or $u_{ii}$ = 1 for $i$ = 1,2, … $n$, both $\boldsymbol{L}$ and $\boldsymbol{U}$ are unique.

**Proof:** For $k$ =1, there is nothing to prove. Let us assume that the theorem is valid up to $k$. We will check for the validity for $\boldsymbol{A_{k+1}}$:

$$A_{k+1} = \begin{bmatrix} A_k & \overline{a} \\ \overline{b}^T & a_{k+1k+1} \end{bmatrix} \qquad (2.57)$$

If this has an $LU$ decomposition, they will be of the form:

$$L_{k+1} = \begin{bmatrix} L_k & 0 \\ \overline{\beta}^T & 1 \end{bmatrix} \text{ and } U_{k+1} = \begin{bmatrix} U_k & \overline{\alpha} \\ 0 & u_{kk} \end{bmatrix} \tag{2.58}$$

By multiplying and equating the elements, we obtain:

$$A_k = L_k U_k \tag{2.59}$$

$$L_k \overline{\alpha} = \overline{a} \tag{2.60}$$

$$\overline{\beta}^T U_k = \overline{b}^T \text{ or } U_k \overline{\beta} = \overline{b} \tag{2.61}$$

$$\overline{\beta}^T \overline{\alpha} + u_{kk} = a_{kk} \tag{2.62}$$

Since the theorem is valid up to $A_k$, decomposition in equation (2.59) is valid and $\det(A_k) \neq 0$. Moreover, determinant of a triangular matrix (upper or lower) is the product of the diagonal entries. Thus, $\det(A_k) = \det(L_k)\det(U_k) \neq 0$. So, equations (2.60) and (2.61) have unique solution vectors, $\overline{\beta}$ and $\overline{\alpha}$. Then, by equation (2.62), $u_{kk}$ is also uniquely determined.

Once such a decomposition is obtained, one can express the equation of the form $\boldsymbol{Ax} = \boldsymbol{b}$ as $\boldsymbol{LUx} = \boldsymbol{b}$. The later is equivalent to two triangular systems of equations: $\boldsymbol{Ux} = \boldsymbol{y}$ and $\boldsymbol{Ly} = \boldsymbol{b}$. One can easily obtain $\boldsymbol{y}$ by solving the triangular system $\boldsymbol{Ly} = \boldsymbol{b}$ by *forward substitution* and subsequently obtain the solution vector $\boldsymbol{x}$ by solving the other triangular system $\boldsymbol{Ux} = \boldsymbol{y}$ by *back substitution*.

In the enlarged from, $LU$ decomposition looks as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ a_{31} & a_{32} & \cdot & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdot & \cdot & 0 \\ l_{21} & l_{22} & 0 & \cdot & 0 \\ l_{31} & l_{32} & l_{33} & \cdot & 0 \\ & \cdot & \cdot & \cdot & \cdot \\ l_{n1} & l_{n2} & l_{n3} & \cdot & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdot & \cdot & u_{1n} \\ 0 & u_{22} & \cdot & \cdot & u_{2n} \\ 0 & 0 & u_{33} & \cdot & u_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & u_{nn} \end{bmatrix} \tag{2.63}$$

Reader should note at this point that for a given $n \times n$ matrix $\boldsymbol{A}$, all $a_{ij}$'s are known. On the right hand side of the equations all $l_{ij}$'s and $u_{ij}$'s are unknown. Since there are $\frac{n}{2}(n+1)$ non-zero elements in a triangular matrix, all total there are $(n^2+n)$ unknowns. By simple matrix multiplication, one can obtain $n^2$ equation since there are $n^2$ numbers of known $a_{ij}$ values. As a result, values of n unknowns have to be set arbitrarily. This is accomplished by setting either $l_{ii}$ or $u_{ii} = 1$ for $i = 1,2, \ldots n$. This allows unique determination of the remaining $n^2$ unknowns.

Theoretically, one can obtain these values by writing these equations explicitly through matrix multiplication and solving them. This is often simpler for a small system, for example a $3 \times 3$ matrix. However, for a large system, this becomes quite cumbersome and a more systematic method is called for. In order to do that, we will use Gauss elimination steps and

algorithm to show that it is equivalent to *LU* decomposition. In the process we will derive the general algorithm for the *LU* decomposition.

Since it is difficult to visualize a $n \times n$ matrix, let us first try to visualize the steps of Gauss elimination with a 5 x 5 matrix :

$$\begin{bmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 & a_{14}^0 & a_{15}^0 \\ a_{21}^0 & a_{22}^0 & a_{23}^0 & a_{24}^0 & a_{25}^0 \\ a_{31}^0 & a_{32}^0 & a_{33}^0 & a_{34}^0 & a_{35}^0 \\ a_{41}^0 & a_{42}^0 & a_{43}^0 & a_{44}^0 & a_{45}^0 \\ a_{51}^0 & a_{52}^0 & a_{53}^0 & a_{54}^0 & a_{55}^0 \end{bmatrix} \xrightarrow{Step\ 1} \begin{bmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 & a_{14}^0 & a_{15}^0 \\ 0 & a_{22}^1 & a_{23}^1 & a_{24}^1 & a_{25}^1 \\ 0 & a_{32}^1 & a_{33}^1 & a_{34}^1 & a_{35}^1 \\ 0 & a_{42}^1 & a_{43}^1 & a_{44}^1 & a_{45}^1 \\ 0 & a_{52}^1 & a_{53}^1 & a_{54}^1 & a_{55}^1 \end{bmatrix} \xrightarrow{Step\ 2}$$

$$\begin{bmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 & a_{14}^0 & a_{15}^0 \\ 0 & a_{22}^1 & a_{23}^1 & a_{24}^1 & a_{25}^1 \\ 0 & 0 & a_{33}^2 & a_{34}^2 & a_{35}^2 \\ 0 & 0 & a_{43}^2 & a_{44}^2 & a_{45}^2 \\ 0 & 0 & a_{53}^2 & a_{54}^2 & a_{55}^2 \end{bmatrix} \xrightarrow{Step\ 3} \begin{bmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 & a_{14}^0 & a_{15}^0 \\ 0 & a_{22}^1 & a_{23}^1 & a_{24}^1 & a_{25}^1 \\ 0 & 0 & a_{33}^2 & a_{34}^2 & a_{35}^2 \\ 0 & 0 & 0 & a_{44}^3 & a_{45}^3 \\ 0 & 0 & 0 & a_{54}^3 & a_{55}^3 \end{bmatrix} \xrightarrow{Step\ 4} \begin{bmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 & a_{14}^0 & a_{15}^0 \\ 0 & a_{22}^1 & a_{23}^1 & a_{24}^1 & a_{25}^1 \\ 0 & 0 & a_{33}^2 & a_{34}^2 & a_{35}^2 \\ 0 & 0 & 0 & a_{44}^3 & a_{45}^3 \\ 0 & 0 & 0 & 0 & a_{55}^4 \end{bmatrix} \quad (2.64)$$

Note that the superscripts represent number of times the value has been actively modified during the step operations according to eq (2.52) in Gauss Elimination. This process can be thought of as a sequence of matrices generated from the original matrix by the steps of Gauss elimination and can be written as:

$$A^{(1)} \xrightarrow{Step\ 1} A^{(2)} \xrightarrow{Step\ 2} A^{(3)} \xrightarrow{Step\ 3} A^{(4)} \xrightarrow{Step\ 4} A^{(5)} \quad (2.65)$$

Furthermore, let's define $A^{(k)} = \left[a_{ij}^{(k)}\right]$ (2.66)

The superscripts in the parentheses in eq (2.65) and (2.66) are different from those in eq (2.64). In equation (2.64), superscripts represent the number of times a non-zero entry has been actively modified during the Gauss elimination steps. While the superscript "(*k*)" in eq. (2.65) and (2.66) corresponds to the elements at the beginning of *Step k* of Gauss elimination (2.52). Let us focus on the entries at or above the main diagonal ( $i \le j$ ) and below the main diagonal ($i > j$) separately, for the number of times they have been modified:

- For $i \le j$ , the entries ($a_{ij}$'s) are actively modified for ($i$-1) steps. For rest of the steps, they remain unaltered, *i.e.,* $a_{ij}^{(i)} = a_{ij}^{(i+1)} = \cdots = a_{ij}^{(n)}$ .

- For $i > j$, the entries ($a_{ij}$'s) are actively modified for $j$ steps. For rest of the steps, they are zero, *i.e.,* $a_{ij}^{(j+1)} = a_{ij}^{(j+2)} = \cdots = a_{ij}^{(n)} = 0$ .

The above two statements are mathematically equivalent to saying that any entry $a_{ij}$ is altered for $m$ steps where, $m = min\ (i\text{-}1, j)$. At each step of the Gauss elimination, the elements are modified according to eq (2.52), which can now be written following the additional superscript convention of eq (2.66), as follows:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)} \text{ where, } l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \tag{2.67}$$

Notice, that the step number corresponds to the pivotal index as in the Gauss elimination. Since, any element is modified for $m$ steps, we can sum eq (2.67) over $m$ steps to get the original elements back.

$$\sum_{k=1}^{m} \left( a_{ij}^{(k+1)} - a_{ij}^{(k)} \right) = -\sum_{k=1}^{m} l_{ik} a_{kj}^{(k)} \text{ or } a_{ij}^{(m+1)} - a_{ij}^{(1)} = -\sum_{k=1}^{m} l_{ik} a_{kj}^{(k)} \tag{2.68}$$

Note that $a_{ij}^{(1)}$'s are the original elements of matrix $\boldsymbol{A}$. Expanding (2.68) for the original elements ($a_{ij}$'s), one can write more specifically for values of "$m$" above and below the diagonal:

$$a_{ij} = \begin{cases} a_{ij}^{(i)} + \sum_{k=1}^{i-1} l_{ik} a_{kj}^{(k)}, & i \le j \\ 0 + \sum_{k=1}^{j} l_{ik} a_{kj}^{(k)}, & i > j \end{cases} \tag{2.69}$$

If we define, $l_{ii} = 1$, we can write (2.69) simply as:

$$a_{ij} = \sum_{k=1}^{t} l_{ik} a_{kj}^{(k)} \text{ where, } t = \min(i, j) \tag{2.70}$$

This is nothing but product of two matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ whose elements are given by,

$$l_{ij} = \begin{cases} l_{ij} \text{ for } i > j \\ 1 \text{ for } i = j \\ 0 \text{ for } i < j \end{cases} \text{ and } u_{ij} = \begin{cases} a_{ij}^{(i)} \text{ for } i \le j \\ 0 \text{ for } i > j \end{cases} \tag{2.71}$$

Thus, Gauss elimination is equivalent to $LU$ decomposition where the final upper triangular matrix is the $\boldsymbol{U}$ matrix and one can construct the $\boldsymbol{L}$ matrix simply by saving the multiplying factor $l_{ik}$ at each step.

The equations (2.69) and (2.70) are two alternative forms of the general formulation for $LU$ decomposition. They provide $n^2$ independent equations for the determination of the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$. It is a small step to derive different forms of $LU$ decomposition algorithms using these general equations, which allows systematic calculation of the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$.

*Doolittle's Algorithm*

In equation (2.69), if we put $u_{kj} = a_{kj}^{(k)}$ and $l_{kk} = 1$, we can rewrite as:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}, j = i, i+1, \cdots n$$
$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}}, i = j+1, j+2 \cdots n \quad (2.72)$$

*Crout's Algorithm*

Define $u_{kj} = a_{kj}^{(k)}$ and $u_{kk} = 1$ in eq (2.69) to obtain:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}, i = j, j+1, \cdots n$$
$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}}{l_{ii}}, j = i+1, i+2 \cdots n \quad (2.73)$$

*Cholesky Algorithm*

The $LU$ decomposition for a symmetric and positive definite matrix $A$ can be computed much more efficiently if we change the composition of matrices $L$ and $U$ somewhat. The decomposition in this case can be expressed as $A = LL^T$. Then one needs to compute the elements of $L$ and the $U$ will be $L^T$. This substantially reduces the computational effort and CPU time. Why we specially require a symmetric and positive definite matrix for this will be clear once we go through the details of the derivation. At this point, let us take a closer look at the $U$ matrix. The upper triangular matrix from $LU$ decomposition can be further decomposed into a diagonal matrix and an upper triangular matrix with diagonal elements as 1. For example:

$$\begin{bmatrix} d_1 & u_{12} & \cdot & \cdot & u_{1n} \\ 0 & d_2 & \cdot & \cdot & u_{2n} \\ 0 & 0 & d_3 & \cdot & u_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdots d_n \end{bmatrix} = \begin{bmatrix} d_1 & 0 & \cdot & \cdot & 0 \\ 0 & d_2 & \cdot & \cdot & 0 \\ 0 & 0 & d_3 & \cdot & 0 \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdots d_n \end{bmatrix} \begin{bmatrix} 1 & u_{12}/d_1 & \cdot & \cdot & u_{1n}/d_1 \\ 0 & 1 & \cdot & \cdot & u_{2n}/d_2 \\ 0 & 0 & 1 & \cdot & u_{3n}/d_3 \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdots 1 \end{bmatrix} \quad (2.74)$$

Thus, instead of $LU$, one can talk about a more general $LDU$ decomposition of a matrix. The statement for existence of such a decompostion is given by theorem 2.4.

**Theorem 2.4:** Let $A$ be a $n \times n$ invertible matrix then there exists a decomposition of the form $A = LDU$ where, $L$ is a $n \times n$ lower triangular matrix with diagonal elements as 1, $U$ is a $n \times n$ upper triangular matrix with diagonal elements as 1, and $D$ is a diagonal matrix.

We leave this proof to the readers

Let us highlight the derivation of Cholesky method with an example. Notice that the diagonal elements of the U matrix, which are originally the pivots of the Gauss elimination

are retained in the diagonal matrix (eq 2.74). Now let us look at the *LDU* decomposition of a symmetric matrix. We use a 3 x 3 matrix for illustration:

$$\begin{bmatrix} d_1 & a & b \\ a & d_2 & c \\ b & c & d_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_2 & 0 \\ 0 & 0 & p_3 \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.75}$$

We can write the $l_{ij}$'s and $u_{ij}$'s either using the Gauss elimination or Doolittle's algorithme and modify the $u_{ij}$'s according to (2.74). These are as follows,

$$\begin{aligned} l_{21} &= u_{12} = \frac{a}{d_1} = \alpha \text{ (say)} \\ l_{31} &= u_{13} = \frac{b}{d_1} = \beta \text{ (say)} \\ l_{32} &= u_{23} = \frac{cd_1 - ab}{d_1 d_2 - a^2} = \gamma \text{ (say)} \\ p_1 &= d_1 \\ p_2 &= \frac{d_1 d_2 - a^2}{d_1} \\ p_3 &= \frac{\left(d_1 d_2 - a^2\right)\left(d_1 d_3 - b^2\right) - \left(d_1 c - ab\right)^2}{\left(d_1 d_2 - a^2\right) d_1} \end{aligned} \tag{2.76}$$

So, the above equation can be written as:

$$\begin{bmatrix} d_1 & a & b \\ a & d_2 & c \\ b & c & d_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ \beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_2 & 0 \\ 0 & 0 & p_3 \end{bmatrix} \begin{bmatrix} 1 & \alpha & \beta \\ 0 & 1 & \gamma \\ 0 & 0 & 1 \end{bmatrix} \tag{2.77}$$

Notice, that in this case $U = L^T$. In fact, if you pay close attention to the process of Gauss elimination, you will notice, this will always be the case for a symmetric matrix. Therefore, for a symmetric matrix, the decomposition can be written as $A = LDL^T$.

The pivots ($p_i$'s) of Gauss elimination are in the diagonal matrix $D$. Since, all pivots are positive for a positive definite matrix, their square roots are real. Square root of a diagonal matrix is a diagonal matrix with the square root of the diagonal elements. Thus, the above decomposition for a positive definite matrix can be written as:

$$A = LDL^T = \left(LD^{1/2}\right)\left(D^{1/2}L^T\right) = L_C L_C^T \tag{2.78}$$

where, $L_C = LD^{1/2}$ (2.79)

This is known as *Cholesky* decomposition. In the above example of 3 x 3 matrix, the decomposition is as follows:

$$\begin{bmatrix} d_1 & a & b \\ a & d_2 & c \\ b & c & d_3 \end{bmatrix} = \begin{bmatrix} \sqrt{p_1} & 0 & 0 \\ \alpha\sqrt{p_1} & \sqrt{p_2} & 0 \\ \beta\sqrt{p_1} & \gamma\sqrt{p_2} & \sqrt{p_3} \end{bmatrix} \begin{bmatrix} \sqrt{p_1} & \alpha\sqrt{p_1} & \beta\sqrt{p_1} \\ 0 & \sqrt{p_2} & \gamma\sqrt{p_2} \\ 0 & 0 & \sqrt{p_3} \end{bmatrix} \tag{2.80}$$

Once, you follow the logic of the Cholesky decomposition described above, the elementwise algorithm can be written easily. We present the algorithm below and leave the reader for checking how it can be derived similar to the *LU* algorithme.

$$k = 1,2,...n$$

$$l_{kk} = \left( a_{kk} - \sum_{p=1}^{k-1} l_{kp}^2 \right)^{\frac{1}{2}}$$

$$l_{ik} = \frac{a_{ik} - \sum_{p=1}^{k-1} l_{ip} l_{kp}}{l_{kk}}, \quad i = k+1,...,n \tag{2.81}$$

**Example 2.7:** Solve the Batman problem of equation (2.2) by performing an *LU* decomposition of the coefficient matrix using, (a) Gaussian Elimination, (b) Doolittle algorithm, (c) Crout Algorithm.

**Solution:**

(a) Gaussian Elimination

The problem was solved by Gauss Elimination in example 2.4. We use those results to assemble the *L* and *U* matrices. Recall that the *U* matrix is the same as the final upper triangular matrix obtained in Gauss Elimination:

$$U = \begin{bmatrix} 1 & 6.2 & 19.22 \\ 0 & 8 & 81.6 \\ 0 & 0 & 66.42 \end{bmatrix}$$

Using the values of $l_{21} = 1$, $l_{31} = 1$ and $l_{32} = 2.025$, we can assemble the L matrix as:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2.025 & 1 \end{bmatrix}$$

Now, we can first solve the equation $Ly = b$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2.025 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 439 \\ 640 \\ 875 \end{bmatrix}$$

Using forward substitution of eq (2.44), we obtain the solution vector *y* as,

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 439 \\ 201 \\ 28.975 \end{bmatrix}$$

Notice, that the above is same as the augmented right hand side vector in example 2.4 after the Gauss Elimination. In order to obtain the solution vector x we now have to solve the following:

$$\begin{bmatrix} 1 & 6.2 & 19.22 \\ 0 & 8 & 81.6 \\ 0 & 0 & 66.42 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 439 \\ 201 \\ 28.975 \end{bmatrix}$$

The above can be solved by the back substitution of eq (2.42). Since, the equations are same as in example 3.1, we obtain the same solutions as, $x_1 = S_0 = 302.42824$, $x_2 = u = 20.67536$ and $x_3 = f = 0.436$.

(b) Doolittle's method:

The coefficient matrix is $A = \begin{bmatrix} 1 & 6.2 & 19.22 \\ 1 & 14.2 & 100.82 \\ 1 & 22.4 & 250.88 \end{bmatrix}$

Now, we evaluate the elements of matrices $L$ and $U$ using the set of equation (2.72). The sequence in which the elements can be evaluated are

$i = 1, j = 1, u_{11} = a_{11} = 1$

$i = 1, j = 2, u_{12} = a_{12} = 6.2$

$i = 1, j = 3, u_{13} = a_{13} = 19.22$

$j = 1, i = 2, l_{21} = 1/1 = 1$

$j = 1, i = 3, l_{31} = 1/1 = 1$

$i = 2, j = 2, u_{22} = 14.2\text{-}(1 \text{ x } 6.2) = 8$

$i = 2, j = 3, u_{23} = 100.82\text{-}(1 \text{ x } 19.22) = 81.6$

$j = 2, i = 3, l_{32} = (22.4 - (1 \text{ x } 6.2))/8 = 2.025$

$i = 3, j = 3, u_{33} = 250.88 - (1 \text{ x } 19.22) - (2.025 \text{ x } 81.6) = 66.42$

Although it may appear from eq (2.72) that the elements of matrix $U$ and $L$ can be calculated independently, it is not so. Notice in the above example, the first row of matrix $U$ is first calculated followed by the first column of $L$. This is important because in calculation of 2nd row of the matrix $U$, the elements of the 1st column of L were required. So, the sequence of computation in Doolittle's method can be presented as:

1st Row of $U \Rightarrow$ 1st Column of $L \Rightarrow$ 2nd Row of $U \Rightarrow$ 2nd Column of $L \Rightarrow$ so on…

Following the computation of the elements, the $L$ and $U$ matrices of the Doolittle method can be synthesized as follows:

$$U = \begin{bmatrix} 1 & 6.2 & 19.22 \\ 0 & 8 & 81.6 \\ 0 & 0 & 66.42 \end{bmatrix} \text{ and } L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2.025 & 1 \end{bmatrix}$$

(c) Crout's method:

The elements of the Crout's decomposition can be calculated using eq (2.73) in the same way as was done in Doolittle's method. However, recall that the difference is that the diagonal elements of $U$ matrix is now unity. The computation sequence is also changed accordingly. The computation sequence of Crout's is:

$1^{st}$ Column of $L \Rightarrow 1^{st}$ Row of $U \Rightarrow 2^{nd}$ Column of $L \Rightarrow 2^{nd}$ Row of $U \Rightarrow$ so on …..

Using the above computation sequence in combination with eq (2.73), the $L$ and $U$ matrices can be computed as:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 8 & 0 \\ 1 & 16.2 & 66.42 \end{bmatrix} \text{ and } U = \begin{bmatrix} 1 & 6.2 & 19.22 \\ 0 & 1 & 10.2 \\ 0 & 0 & 1 \end{bmatrix}$$

Using the $LU$ decompositions in (b) and (c), the solutions can be computed similarly as was done in the case of (a).

**Example 2.8:** Perform a Cholesky decomposition of the following positive definite matrix.

$$\begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 1 & 4.3125 \end{bmatrix}$$

**Solution:** Using eq (2.81), the elements of matrix $L$ can be computed as follows:

$k = 1,\ l_{11} = \sqrt{4} = 2$

$k = 1,\ i = 2,\ l_{21} = 2/2 = 1$

$k = 1,\ i = 3,\ l_{31} = 1/2 = 0.5$

$k = 2,\ l_{22} = \sqrt{4} = 2$

$k = 2,\ i = 3,\ l_{32} = (1 - (1 \text{ x } 0.5))/2 = 0.25$

$k = 3,\ l_{33} = \sqrt{4.3125 - 0.5^2 - 0.25^2} = 2$

So, the matrix L can be synthesized as: $L = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 0.5 & 0.25 & 2 \end{bmatrix}$

The Cholesky decomposition is then given by $A = LL^T$ as follows:

$$\begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 1 & 4.3125 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 0.5 & 0.25 & 2 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0.5 \\ 0 & 2 & 0.25 \\ 0 & 0 & 2 \end{bmatrix}$$

### 2.2.1.4 Banded Matrices and Thomas Algorithme

Advantage of the direct methods described so far is that the solution can be obtained in finite number of steps and as a result the quantum of round-off error can be estimated. However, if the co-efficient matrix contains a lot of zeroes as elements, it is easy to see that many computations will involve operation on zero. It is too cumbersome to eliminate these computations on zeroes within the framework of the methods described above. At the same time, one wishes to eliminate the computations involving zeroes thereby making the solution procedure more efficient. Before proceeding with the methods, let us categorize a few types of matrices based on the zeroes and their locations in the matrix.

A matrix will be called *Dense* if most of the elements are non-zero. Alternatively, if the matrix contains a large proportion of the elements as zero, it will be called a *Sparse* matrix. A special kind of *Sparse* matrix contains non-zero elements only along the diagonal and lines parallel to the diagonal elements, both above and below the diagonal of a matrix. These are known as *Banded* matrix and one such is shown in Fig. 2.5. *Band width* ($s$) of a banded matrix shown in Fig. 2.5 is $s = (a + b - 1)$. So, for a banded matrix $a_{ij} = 0$ for $j > i + a$ or $i > j + b$. A special case is when $a = b = 2$ and band width is $s = 3$, the matrix is termed as a tri-diagonal.

$$\begin{array}{c} \leftarrow a \rightarrow \\ \uparrow \\ \\ b \\ \downarrow \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} \times & \times & \times & 0 & 0 & 0\ldots & 0 \\ \times & \times & \times & \times & 0 & 0\ldots & 0 \\ \times & \times & \times & \times & \times & 0\ldots & 0 \\ \times & \times & \times & \times & \times & \times\ldots & 0 \\ 0 & \times & \times & \times & \times & \times\ldots & 0 \\ 0 & 0 & \times & \times & \times & \times\ldots & 0 \\ . & . & . & . & . & . & . \end{bmatrix}$$

**Figure 2.5 A banded matrix of band width (a + b – 1). An "x" denotes positions that may contain non-zero entries.**

*Thomas Algorithm for Tridiagonal Matrix*

This special kind of sparse matrix often appear in the computation as a result of finite difference solution of differential equations in one-dimensional problems (and sometimes, in 2-diemsional problems with special algorithms). Solution of these matrices can be accomplished using the direct methods described before where most of the computations would yield zero. However, instead of matrix operation, a tridiagonal system can be solved by vector operations. Thomas algorithm is one such procedure. Let's consider the following tridiagonal matrix:

$$\begin{bmatrix} d_1 & u_1 & 0 & \bullet & 0 & 0 \\ l_2 & d_2 & u_2 & \bullet & 0 & 0 \\ 0 & l_3 & d_3 & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & l_{n-1} & d_{n-1} & u_{n-1} \\ 0 & 0 & 0 & 0 & l_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \bullet \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \bullet \\ b_{n-1} \\ b_n \end{bmatrix} \tag{2.82}$$

The above matrix equation can be stored in four vectors of length $n$. These are vector $\boldsymbol{d}$ consisting of the diagonal elemens, vector $\boldsymbol{u}$ consisting of the upper diagonal elements, vector $\boldsymbol{l}$ consisting of the lower diagonal elements and the right hand side vector $\boldsymbol{b}$. The $i^{\text{th}}$ equation can be written as follows:

$$l_i x_{i-1} + d_i x_i + u_i x_{i+1} = b_i \tag{2.83}$$

Note that $l_1$ and $u_n$ are zero. Let us define two new vectors, $\alpha$ and $\beta$ of length $n$ and initialize them as $\alpha_1 = d_1$ and $\beta_1 = b_1$. Using these, the first two equations can be written as follows:

$$\begin{aligned} \alpha_1 x_1 + u_1 x_2 &= \beta_1 \\ l_2 x_1 + d_2 x_2 + u_2 x_3 &= b_2 \end{aligned} \tag{2.84}$$

Multiplying the first equation by $\dfrac{l_2}{\alpha_1}$ and deducting from the second equation, we get,

$$\alpha_2 x_2 + u_2 x_3 = \beta_2 \text{ where } \alpha_2 = d_2 - \left(\frac{l_2}{\alpha_1}\right) u_1 \text{ and } \beta_2 = b_2 - \left(\frac{l_2}{\alpha_1}\right) \beta_1 \tag{2.85}$$

Similarly, we can subsequently remove $x_2$, $x_3$ and so on. Note that the vectors $\alpha$ and $\beta$ generated during the process of elimination can simply be calculated by the iterative formulae:

$$\alpha_i = d_i - \left(\frac{l_i}{\alpha_{i-1}}\right) u_{i-1} \text{ and } \beta_i = b_i - \left(\frac{l_i}{\alpha_{i-1}}\right) \beta_i \text{ , } i = 2,\ldots..n. \tag{2.86}$$

At the $i$th stage, the resulting equation can be written as

$$\alpha_i x_i + u_i x_{i+1} = \beta_i \tag{2.87}$$

After the completion of the elimination steps, the last equation becomes $\alpha_n x_n = \beta_n$ which leads to the solution $x_n = \frac{\beta_n}{\alpha_n}$. All other $x_i$'s can easily be calculated as follows:

$$x_i = \frac{\beta_i - u_i x_{i+1}}{\alpha_i}, \; i = n\text{-}1, \ldots..1. \tag{2.88}$$

**Example 2.9:** Solve the following tridiagonal system of equation using Thomas Algorithm.

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -4 & 1 & 0 \\ 0 & 1 & -4 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 2 \\ -2 \end{bmatrix}$$

**Solution:** Using the same notation for the lower and upper diagonal and the diagonal elements as in eq (2.82), we can compute $\alpha_i$'s and $\beta_i$'s using eq (2.86). Finally the solution vector can be computed using eq (2.88). We tabulate the computations below:

| ***index*** | ***l*** | ***d*** | ***u*** | ***b*** | ***α*** | ***β*** | ***x*** |
|---|---|---|---|---|---|---|---|
| 1 | | -2 | 1 | 3 | -2 | 3 | -1.93333 |
| 2 | 1 | -4 | 1 | 1 | -3.5 | 2.5 | -0.86667 |
| 3 | 1 | -4 | 1 | 2 | -3.71429 | 2.714286 | -0.53333 |
| 4 | 1 | -2 | | -2 | -1.73077 | -1.26923 | 0.733333 |

### 2.2.1.5 Stability of the Direct methods and Pivoting

In the context of Gauss elimination, what we mean by stability is that the round-off errors do not grow. Round-off error incurred in any element will grow only if it is multiplied by a number of magnitude larger than unity at any time during the process of elimination. Looking at the process of elimination, it is easy to see that at each step, elements are multiplied by the factor $l_{ik}$. So, the condition of stability is:

$$|l_{ik}| \leq 1 \tag{2.89}$$

Since, $l_{ik} = \frac{a_{ik}}{a_{kk}}$ for $i = (k + 1)$ to $n$ if follows that $|a_{kk}| \geq |a_{ik}| \; \forall i = (k+1) \cdots n$. In other words, at any step $k$, the absolute value of the pivot should be larger than the absolute values of the elements in the pivotal column of the subsequent rows.

If at any step, the above condition is not satisfied, it can be easily achieved by interchanging of rows. In order to illustrate this, let's have a re-look at the Gauss Elimination matrix in the beginning of the $k^{th}$ step:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1k} & a_{1k+1} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & a_{24} & \cdots & a_{2k} & a_{2k+1} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & a_{34} & \cdots & a_{3k} & a_{3k+1} & \cdots & a_{3n} \\ 0 & 0 & 0 & a_{44} & \cdots & a_{4k} & a_{4k+1} & \cdots & a_{4n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & a_{kk} & a_{kk+1} & \cdots & a_{kn} \\ 0 & 0 & 0 & 0 & \cdots & a_{k+1k} & a_{k+1k+1} & \cdots & a_{k+1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & a_{nk} & a_{nk+1} & \cdots & a_{nn} \end{bmatrix} \tag{2.90}$$

Before we perform the elimination operations of the $k^{th}$ step, we search for $\max_{k \le i \le n} |a_{ik}|$. If this occurs at $i = p$, we interchange $k^{th}$ row with the $p^{th}$ row (of course, accompanied by the interchange of entries in the right hand side vector as well, i.e., interchange $b_k$ with $b_p$. This is done automatically if one works with the augmented matrix.) It is equivalent to changing the position or sequence of the equations in the set and thus has no bearing on the final solution. The operation is known as *partial pivoting*.

You may have already guessed that there is also something called the *full pivoting*. In this, the search is made for the largest absolute valued element in rows as well as columns between $k$ and $n$, that is, search for $\max_{\substack{k \le i \le n \\ k \le j \le n}} |a_{ij}|$. If this occurs in the $p^{th}$ row and $s^{th}$ column, interchange $k^{th}$ row with the $p^{th}$ row and $k^{th}$ column with $s^{th}$ column such that $a_{ps}$ becomes the pivot. While partial pivoting had no bearing on the final solution, exchange of columns in full pivoting amounts to renaming of the variable. So, this should be properly accounted for while interpreting the final solution vector.

We have already shown that the *LU* decomposition is nothing but Gauss elimination. So, the same stability criteria and pivoting scheme are applicable there as well. It is left to the reader to identify the stability criteria and design partial pivoting scheme for Gauss-Jordon and Thomas algorithm.

**Example 2.10**: Perform Gauss Elimination on the matrix $\begin{bmatrix} 2 & 1 & -1 \\ 1 & -0.5 & -1 \\ 2 & 4 & 1 \end{bmatrix}$

**Solution:** After the 1st step of Gauss Elimination, the matrix is $\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0 & -0.5 \\ 0 & 3 & 2 \end{bmatrix}$.

We run into a problem to proceed further with elimination as the pivot $a_{22} = 0$. However, if we switch the 3rd row with the 2nd, we obtain,

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 3 & 2 \\ 0 & 0 & -0.5 \end{bmatrix}$$

This is already an upper triangular matrix.

### 2.2.1.6 Perturbation Analysis

In many engineering applications, one is interested in finding out the effect of small changes in some elements of the coefficient matrix and/or in the elements of the right hand side vector on the solution vector. These small changes in the coefficient matrix or in the right hand side vector often represent small perturbations in the physical process. In this section, we will try to establish a relation between the perturbation quantities and the resulting changes in the solution vector. More precisely, we would like to obtain upper bounds for the changes in the solution vector as a function of the perturbation quantities. This analysis is immensely important from the application point of view where one often is interested in the possibility of failure as a result of small perturbation. An analysis of upper bounds of the effect will enable an engineer to restrict the perturbations within certain limits to avoid failure.

We first explore the effect of perturbation in the co-efficient matrix $A$. Let us assume that the matrix $A$ has been given a perturbation of $\delta A$. As a result the solution vector changes by an amount $\delta x$. Notice that if the matrix A is a $(n \times n)$ matrix, $\delta A$ is also one with at least one non-zero element which is small compared to the corresponding element in $A$. If more than one element in $\delta A$ are non-zero, all such elements are small compared to the corresponding elements in $A$. So, the resulting equation to solve now is as follows:

$$(A + \delta A)(x + \delta x) = b \tag{2.91}$$

Since, $Ax = b$, the above equation can be simplified to yield the following:

$$A.\delta x + \delta A(x + \delta x) = 0 \text{ or } \delta x = -A^{-1}\delta A(x + \delta x) \tag{2.92}$$

Now, we take the norm in order to get the upper bound. Notice that the negative sign disappears due to this.

$$\|\delta x\| = \|A^{-1}\delta A(x + \delta x)\| \le \|A^{-1}\|\|\delta A\|\|x + \delta x\| \le \|A^{-1}\|\|\delta A\|\|x\| + \|A^{-1}\|\|\delta A\|\|\delta x\| \tag{2.93}$$

Second term on the right hand side involve product of two perturbation quantities which is likely to be much smaller compared to the first term. So, ignoring this term and carrying out some algebraic manipulation, one obtains,

$$\frac{\|\delta x\|}{\|x\|} = \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|} \tag{2.94}$$

If we did not ignore the product term, the relationship would be,

$$\frac{\|\delta x\|}{\|x+\delta x\|} = \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|} \tag{2.95}$$

Notice that both the equations (2.94) and (2.95) are essentially saying the same thing, that is, the relative change in vector $x$ is proportional to the relative change in the matrix $A$ with the proportionality constant $\|A\|\|A^{-1}\|$. One expresses the relative change in $x$ with respect to the old vector $x$ and the other with respect to the new vector $(x + \delta x)$. Since, the perturbation quantity is likely to be smaller compared to the original vector, these ratios are unlikely to make much difference. In other words, ignoring the term involving product of two perturbation quantities in eq (2.93) does not have much effect on the final results. One may use either of the two equations. The proportionality constant $\|A\|\|A^{-1}\|$ is known as the *condition number* of the matrix $A$. We will denote the condition number as $C(A)$. It is easy to see, that if $C(A)$ is very large, the error is amplified considerably and for small $C(A)$, the error stays within acceptable limits. Depending on this, the matrix can be categorized as *ill conditioned* if $C(A)$ is large and *well conditioned* otherwise. How large is large depends on the type of problem and the matrix norm chosen to evaluate the condition number.

Now, let us apply a small perturbation $\delta b$ to the right hand side vector $b$. Let us assume that the resulting change in the solution vector $x$ as $\delta x$. The equation set can now be written as:

$$A(x+\delta x) = (b+\delta b) \tag{2.96}$$

Once again using the equality $Ax = b$, one obtains,

$$\delta x = A^{-1}\delta b \tag{2.97}$$

Taking the norms, we get,

$$\|\delta x\| \le \|A^{-1}\|\|\delta b\| \tag{2.98}$$

Multiplying the numerator and denominator of the right hand side by the norm of vector $b$,

$$\|\delta x\| \le \|A^{-1}\|\frac{\|b\|}{\|b\|}\|\delta b\| \tag{2.99}$$

Using the equality $Ax = b$,

$$\|\delta x\| \le \|A^{-1}\|\frac{\|Ax\|}{\|b\|}\|\delta b\| \le \|A^{-1}\|\|A\|\|x\|\frac{\|\delta b\|}{\|b\|} \tag{2.100}$$

$$\frac{\|\delta x\|}{\|x\|} \le \|A^{-1}\|\|A\|\frac{\|\delta b\|}{\|b\|} \tag{2.101}$$

Once again we observe that relative change in the norms of solution vector $x$ is proportional to the relative perturbation in the norms of the vector $b$ and the proportionality constant is the *condition number* of matrix $A$. Thus, it is safe to say that the sensitivity of the solution to

perturbation in either the coefficient matrix $A$ or the right hand side vector $b$ depend on the condition number of matrix $A$.

**Example 2.11**: In the Batman Problem of (2.2), if the distances were measured with an accuracy of +/- 0.1 m what is the maximum relative error in the estimation of parameters ?

**Solution:** Use $L_1$ norm for the vectors and column sum norm for the matrices, we get:

$\|\delta b\| = 0.1,\ \|b\| = 875,\ \|A\|_1 = 370.92$

Using Gauss-Jordon Elimination, we can compute the inverse as:

$$A^{-1} = \begin{bmatrix} 2.454 & -2.117 & 0.663 \\ -0.282 & 0.436 & -0.154 \\ 0.015 & -0.03 & 0.015 \end{bmatrix}$$

The column sum norm of the inverse is given by the sum of the absolute values of the first column as $\|A^{-1}\|_1 = 2.751$. The condition number of the matrix $A$ is 370.92 x 2.751 = 1020.4 and the upper bound of the relative error in parameter estimation is $\frac{0.1 \times 1020.4}{875} = 0.1166$ or 11.66%.

### 2.2.1.7 Iterative Improvement of Solution by Direct Methods

In the cases of moderate *ill-conditioning*, an iteration scheme can be applied to improve the solution using the direct methods. In order to see this, let us assume that we have computed an approximate solution vector $\tilde{x}$ by a direct method. We can then compute a residual vector $r$,

$$r = b - A\tilde{x} \tag{2.102}$$

The residual vector gives the error vector with the approximate solution. If $x$ is the true solution vector, ($Ax$-$b$) is zero. Adding ($Ax$–$b$) to the right hand side of the above equation, we obtain,

$$A(x - \tilde{x}) = r \tag{2.103}$$

Define $(x - \tilde{x}) = \delta x$ as the improvement in solution. The vector $\delta x$ can be calculated by solving the above equation. However, since this will also not be calculated exactly with infinite precision, an improvement of the existing solution can be calculated as $(\tilde{x} + \delta x)$. Using this improved approximate solution a new residual vector can be calculated. Thus, the equations (2.102) and (2.103) in combination defines an iterative process to improve an approximate solution. Since, the coefficient matrix $A$ remains constant, application of $LU$ decomposition makes the process more efficient. The only thing to keep in mind here is that since the residual vector is a difference of two almost equal quantities, it should be computed with a higher precision. All other quantities, however, may be computed with normal precision, thus saving storage space. If $r$ is not computed using higher precision, there is no gain. On the other hand, if the whole computation is done with higher precision, it significantly increases storage as well as CPU time. Iterative improvement is fast and less computation intensive if the $LU$ decomposition of the matrix $A$ is computed once.

### 2.2.2 *Iterative Methods*

Any $n \times n$ matrix $\boldsymbol{A}$ can be written as a summation of three matrices as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ a_{31} & a_{32} & \cdot & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdot & \cdot & 0 \\ a_{21} & 0 & 0 & \cdot & 0 \\ a_{31} & a_{32} & 0 & 0 & 0 \\ & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & a_{n3} & \cdot & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & \cdot & 0 & 0 \\ 0 & a_{22} & 0 & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & a_{nn} \end{bmatrix} +$$

$$\begin{bmatrix} 0 & a_{12} & a_{13} & \cdot & a_{1n} \\ 0 & 0 & a_{23} & \cdot & a_{2n} \\ 0 & 0 & 0 & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & 0 \end{bmatrix}$$

In the matrix form, this can be written as:

$$\boldsymbol{A = L + D + U} \tag{2.104}$$

We define a *strictly lower triangular matrix* as the one that contains non-zero values below the leading diagonal. All the entries on the diagonal and above the diagonal are zero. Notice that $\boldsymbol{L}$ is a *strictly lower triangular matrix*. A *diagonal matrix* is the one where all entries other than the leading diagonal are zero. The $\boldsymbol{D}$ is a diagonal matrix. Similarly, a *strictly upper triangular matrix* is the one that has non-zero values above the leading diagonal. All the entries on and below the diagonal are zero. The $\boldsymbol{U}$ is such a matrix. Using (2.104), a set of linear simultaneous equation given by $\boldsymbol{Ax = b}$ can be written as follows:

$$\boldsymbol{(L + D + U)\, x = b} \tag{2.105}$$

Recall that if $a_{ij}$'s are the elements of an $n \times n$ matrix $\boldsymbol{A}$, in the expanded form the set of equation is still represented as:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1,\ 2,\ \cdots\ n\,. \tag{2.106}$$

From (2.105), we can derive two iterative methods by algebraic rearrangements. These methods are outlined below.

<u>*Jacobi Iteration*</u>

For an iteration counter $k$,

$$Dx^{(k+1)} = -(U + L)x^{(k)} + b \ \text{ or } \ x^{(k+1)} = -D^{-1}(U + L)x^{(k)} + D^{-1}b \tag{2.107}$$

In the expanded form, it is equivalent to:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1,\ 2,\ \cdots\ n \tag{2.108}$$

*Gauss Seidel Iteration*

During a Jacobi iteration step, some of the $x_i$ values were known in the new iteration step but they were not used in calculation, *e.g.,* $x_1^{(k+1)}$ was known while calculating $x_2^{(k+1)}$ but $x_1^{(k)}$ was used in computation. To make convergence faster, Gauss Seidel iteration scheme uses the known new iteration values of the $\boldsymbol{x}$ vector. For an iteration counter $k$, the scheme is given as follows:

$$(L + D)x^{(k+1)} = -Ux^{(k)} + b \text{ or } x^{(k+1)} = -(L + D)^{-1}Ux^{(k)} + (L + D)^{-1}b \tag{2.109}$$

In the expanded form, it is equivalent to:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1,\ 2,\ \cdots\ n \tag{2.110}$$

Both the schemes require an initial $\boldsymbol{x}$ vector to be given for start-up. It is a common practice to give a zero vector for start-up. However, often in engineering problems, the variables have some physical significance and one knows the order of magnitude values of the variables. This information can be put in the start-up vector to make the convergence faster.

Both the schemes also require a termination criterion in order to be able to stop the iteration. If the true solution vector is $x$ and the approximate solution vector at the $k^{\text{th}}$ iteration is $x^{(k)}$, the true error vector is given by $e^{(k)} = x - x^{(k)}$. However, since the true solution is not known, we will define the approximate error as the improvement between iterations, i.e., $\varepsilon^{(k)} = x^{(k+1)} - x^{(k)}$. An upper bound or tolerance can now be defined on the norm of the error vector $\varepsilon^{(k)}$. Any vector norm defined in section 1.4 can be used for this purpose. Similarly, one can also work with approximate relative error. The error expressions for true and relative errors are $\varepsilon^{(k)} = \left\| x^{(k+1)} - x^{(k)} \right\|$ and $\varepsilon_r^{(k)} = \dfrac{\left\| x^{(k+1)} - x^{(k)} \right\|}{\left\| x^{(k+1)} \right\|}$, respectively.

**Example 2.12**: Solve the Batman problem of (2.2) using (a) Gauss Seidel method and (b) Jacobi method with a precision of 0.001 or less.

**Solution:**

(a) Gauss Seidel Method: We assume an initial solution vector as,

$$\begin{bmatrix} S_0 \\ u \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Now, using equation (2.110), we can compute the new solutions. We tabulate the values below:

| **Iteration No.** | $S_0$ | $u$ | $f$ | $\varepsilon^{(k)}$ |
|---|---|---|---|---|
| | 0 | 0 | 0 | |
| 1 | 439 | 14.15493 | 0.47405 | 439 |
| 2 | 342.1282 | 17.61113 | 0.551588 | 96.8718 |
| 3 | 319.2094 | 18.67461 | 0.547988 | 22.91876 |
| 4 | 312.6851 | 19.15963 | 0.530689 | 6.52436 |
| 5 | 310.0105 | 19.47081 | 0.513566 | 2.674627 |
| 6 | 308.4102 | 19.70507 | 0.499028 | 1.600213 |
| 7 | 307.2372 | 19.8909 | 0.487112 | 1.173016 |
| 8 | 306.3141 | 20.04051 | 0.477433 | 0.923099 |
| 9 | 305.5726 | 20.16145 | 0.469591 | 0.741564 |
| 10 | 304.9735 | 20.25932 | 0.46324 | 0.599104 |
| 11 | 304.4887 | 20.33855 | 0.458098 | 0.484749 |
| 12 | 304.0963 | 20.40269 | 0.453936 | 0.392382 |
| 13 | 303.7787 | 20.45461 | 0.450566 | 0.31765 |
| 14 | 303.5215 | 20.49665 | 0.447838 | 0.257158 |
| 15 | 303.3133 | 20.53068 | 0.445629 | 0.208188 |
| 16 | 303.1448 | 20.55823 | 0.443841 | 0.168544 |
| 17 | 303.0083 | 20.58054 | 0.442393 | 0.136449 |
| 18 | 302.8979 | 20.59859 | 0.441221 | 0.110465 |
| 19 | 302.8084 | 20.61321 | 0.440273 | 0.08943 |
| 20 | 302.736 | 20.62505 | 0.439505 | 0.0724 |
| 21 | 302.6774 | 20.63463 | 0.438883 | 0.058613 |
| 22 | 302.63 | 20.64238 | 0.438379 | 0.047452 |
| 23 | 302.5916 | 20.64866 | 0.437972 | 0.038416 |
| 24 | 302.5605 | 20.65375 | 0.437642 | 0.0311 |
| 25 | 302.5353 | 20.65786 | 0.437375 | 0.025178 |
| 26 | 302.5149 | 20.6612 | 0.437158 | 0.020384 |
| 27 | 302.4984 | 20.66389 | 0.436983 | 0.016502 |
| 28 | 302.485 | 20.66608 | 0.436842 | 0.01336 |
| 29 | 302.4742 | 20.66784 | 0.436727 | 0.010816 |
| 30 | 302.4655 | 20.66928 | 0.436634 | 0.008756 |
| 31 | 302.4584 | 20.67044 | 0.436559 | 0.007089 |
| 32 | 302.4526 | 20.67137 | 0.436498 | 0.005739 |
| 33 | 302.448 | 20.67213 | 0.436449 | 0.004646 |
| 34 | 302.4442 | 20.67275 | 0.436409 | 0.003761 |
| 35 | 302.4412 | 20.67325 | 0.436376 | 0.003045 |
| 36 | 302.4387 | 20.67365 | 0.43635 | 0.002465 |
| 37 | 302.4367 | 20.67397 | 0.436329 | 0.001996 |
| 38 | 302.4351 | 20.67424 | 0.436312 | 0.001616 |
| 39 | 302.4338 | 20.67445 | 0.436298 | 0.001308 |
| 40 | 302.4327 | 20.67463 | 0.436287 | 0.001059 |
| 41 | 302.4319 | 20.67477 | 0.436278 | 0.000857 |

So, the solution vector is,

$$\begin{bmatrix} S_0 \\ u \\ f \end{bmatrix} = \begin{bmatrix} 302.4319 \\ 20.67477 \\ 0.436278 \end{bmatrix}$$

You may want to compare this solution with the one obtained using Gauss Elimination in Example 2.4.

(b) Jacobi Iteration: If we start with the same initial solution vector and perform the iterations using (2.108), we will see that the method does not converge for this problem. The solution oscillates and the oscillation increases with iterations. We tabulate the first 25 iterations below.

| Iteration No. | $S_0$ | $u$ | $f$ | $\varepsilon^{(k)}$ |
|---|---|---|---|---|
| | 0 | 0 | 0 | |
| 1 | 439 | 45.07042 | 3.487723 | 439 |
| 2 | 92.52934 | -10.6079 | -2.28626 | 346.4707 |
| 3 | 548.71097 | 54.78673 | 4.066038 | 456.1816 |
| 4 | 21.172992 | -22.4401 | -3.59109 | 527.538 |
| 5 | 647.14926 | 69.07614 | 5.406906 | 625.9763 |
| 6 | -93.19279 | -38.8925 | -5.25931 | 740.342 |
| 7 | 781.21737 | 88.97437 | 7.331732 | 874.4102 |
| 8 | -253.557 | -62.0002 | -7.57033 | 1034.774 |
| 9 | 968.90277 | 116.6759 | 10.03412 | 1222.46 |
| 10 | -477.2462 | -94.4044 | -10.7918 | 1446.149 |
| 11 | 1231.7256 | 155.301 | 13.81898 | 1708.972 |
| 12 | -789.4667 | -139.786 | -15.2881 | 2021.192 |
| 13 | 1599.507 | 209.2119 | 19.11537 | 2388.974 |
| 14 | -1225.511 | -203.29 | -21.5675 | 2825.018 |
| 15 | 2113.9254 | 284.5032 | 26.52347 | 3339.436 |
| 16 | -1834.701 | -292.114 | -30.3404 | 3948.626 |
| 17 | 2833.2502 | 389.6915 | 36.88241 | 4667.951 |
| 18 | -2685.967 | -416.319 | -42.5994 | 5519.217 |
| 19 | 3838.9405 | 536.6788 | 51.36527 | 6524.908 |
| 20 | -3875.649 | -589.971 | -59.7319 | 7714.59 |
| 21 | 5244.8676 | 742.1002 | 71.61192 | 9120.517 |
| 22 | -5538.402 | -832.731 | -83.6771 | 10783.27 |
| 23 | 7210.2066 | 1029.206 | 99.91461 | 12748.61 |
| 24 | -7862.437 | -1172.08 | -117.145 | 15072.64 |
| 25 | 9957.4566 | 1430.495 | 139.4775 | 17819.89 |

In the next section, we will try to understand under what condition the iterative methods do not converge.

### 2.2.2.1 Convergence of the Iterative Methods

From equations (2.107) and (2.109), it is easy to see that both the methods can be expressed as:

$$x^{(k+1)} = Sx^{(k)} + c \tag{2.111}$$

where, the matrix $\boldsymbol{S}$ and the vector $\boldsymbol{c}$ are given as follows:

For *Jacobi* iteration: $\boldsymbol{S} = -\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})$ and $\boldsymbol{c} = \boldsymbol{D}^{-1}\boldsymbol{b}$ (2.112)

For *Gauss-Seidel* iteration: $\boldsymbol{S} = -(\boldsymbol{L}+\boldsymbol{D})^{-1}\boldsymbol{U}$ and $\boldsymbol{c} = (\boldsymbol{L}+\boldsymbol{D})^{-1}\boldsymbol{b}$ (2.113)

In both the methods, $\boldsymbol{S}$ and $\boldsymbol{c}$, as well as form of the iteration equation (2.111) remain unchanged for all the iterations during the entire solution process. For this reason, these are known as *stationary iteration methods* and the matrix $\boldsymbol{S}$ is called the *iteration matrix*. Note that for a coefficient matrix $\boldsymbol{A}$ of size $n \times n$, the iteration matrix $\boldsymbol{S}$ is also $n \times n$ and $\boldsymbol{c}$ is a vector of size $n$.

If we denote the true solution by $\boldsymbol{x}$, error vector $\boldsymbol{e}^{(k)}$ at $k^{\text{th}}$ iteration is given by:

$$\boldsymbol{e}^{(k)} = \boldsymbol{x} - \boldsymbol{x}^{(k)} \tag{2.114}$$

Needless to say that the true solution vector satisfies the iteration equation (2.111), that is,

$$\boldsymbol{x} = \boldsymbol{S}\boldsymbol{x} + \boldsymbol{c} \tag{2.115}$$

Deducting equation (2.111) from (2.115) and using the definition of (2.114), one obtains:

$$\boldsymbol{e}^{(k+1)} = \boldsymbol{S}\boldsymbol{e}^{(k)} \tag{2.116}$$

The above equation means that the error vector satisfies the homogenous form of the iteration equation (2.111). If at the start of the iteration process, the error vector is given by $\boldsymbol{e}^{(0)}$, error at $k^{\text{th}}$ iteration will be given by,

$$\boldsymbol{e}^{(k)} = \boldsymbol{S}^{\boldsymbol{k}}\boldsymbol{e}^{(0)} \tag{2.117}$$

Thus, the methods will converge if the error vector approaches zero with the progression of the iteration, *i.e.,*

$$\lim_{k \to \infty} e^{(k)} = 0 \tag{2.118}$$

Since $\boldsymbol{e}^{(0)}$ is a constant vector, it is easy to see from (2.117), that (2.118) is possible only if,

$$\lim_{k \to \infty} S^{k} = 0 \tag{2.119}$$

Let us assume that the original linear system of equations and the corresponding coefficient matrix ($\boldsymbol{A}$) is such that the iteration matrix $\boldsymbol{S}$ has a set of linearly independent eigenvectors, which form a basis for an $n$-dimensional vector space. If we denote the eigenvectors of $\boldsymbol{S}$ as $\left\{v_j\right\}_{j=1}^{n}$ and the eigenvalues as $\left\{\lambda_j\right\}_{j=1}^{n}$, any $n$-dimensional vector can be expressed as a linear combination these vectors.

$$\text{If } e^{(0)} = \sum_{j=1}^{n} C_j v_j \tag{2.120}$$

Then, using (2.117), we can write, $e^{(k)} = \sum_{j=1}^{n} C_j \lambda_j^k v_j$

From the above equation one can easily see that in order to satisfy the convergence criterion of (2.118), the largest eigenvalue or the spectral radius (section 2.1.3) of the iteration matrix $S$ must be strictly less than unity, *i.e.,*

$$\rho(\boldsymbol{S}) < 1 \tag{2.121}$$

This is the necessary condition for the methods to converge. However, this condition is very hard to use in application. It will be easier, if we can derive a condition based on the original coefficient matrix $\boldsymbol{A}$. For this purpose, we recall the relation derived in (2.37), $\rho(A) \le \|A\|$ for arbitrary matrix $A$. Using this relation in conjunction with the necessary condition of eq. (2.121), one can get a sufficient condition as,

$$\|S\| < 1 \tag{2.122}$$

Now, for *Jacobi* iteration: $\boldsymbol{S} = -\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})$. The elements ($s_{ij}$) of the iteration matrix are related to $a_{ij}$'s and can be written as follows:

$$s_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}} & \text{for } i \neq j \\ \\ 0 & \text{for } i = j \end{cases} \tag{2.123}$$

If we use, *row-sum norm* (eq. 2.34) for the iteration matrix,

$$\|S\| = \max_{1 \le i \le n} \sum_{j=1}^{n} |s_{ij}| = \max_{1 \le i \le n} \sum_{j=1}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| \tag{2.124}$$

Application of the sufficient condition (2.122) translates to,

$$|a_{ii}| > \sum_{j=1, j \neq i}^{n} |a_{ij}|, \quad i = 1,2,\cdots n \tag{2.125}$$

This means that the *Jacobi* method surely converges if $\boldsymbol{A}$ is strictly diagonally dominant. Using the definition of the iteration matrix in (2.113) and expanded form of (2.110), one can derive a similar condition for the *Gauss-Seidel* method. This is left for the reader as an exercise.

### 2.2.2.2 Rate of Convergence of the iterative methods

From (2.120), for large $k$, one can write:

$$\frac{\left|e^{(k+1)}\right|}{\left|e^{(k)}\right|} \cong \rho(S) \tag{2.126}$$

One is often interested to know how many iterations it would take to reduce the error by a factor of $10^{-m}$. Since $\frac{\left|e^{(k)}\right|}{\left|e^{(0)}\right|} \cong \rho(S)^k$, mathematically the problem boils down to finding the *minimum* integer $k$ for which the equality $\rho(S)^k = 10^{-m}$ is satisfied. The solution of course is $k = -\frac{m}{\log_{10}\rho(S)}$. Notice that the convergence criterion requires $\rho(S)$ to be less than unity which means the right hand side is always positive but unlikely to be an integer. Therefore, the fraction always have to be rounded to the nearest higher integer value to obtain the minimum iteration number. So, instead of equality, it is more appropriate to write it in the form of inequality as,

$$k \geq -\frac{m}{\log_{10}\rho(S)} \tag{2.127}$$

The quantity $R = \log_{10}\rho(S)$ is called the asymptotic rate of convergence. It is also evident that $(1/R)$ represents the minimum number of iteration required to reduce the error by one order of magnitude.

If the eigenvalue corresponding to the spectral radius is denoted as $\lambda_{\max}$ such that, $\rho = \left|\lambda_{\max}\right|$, one may write (2.126) as,

$$e^{(k+1)} \cong \lambda_{\max} e^{(k)} \quad \text{or} \quad e^{(k+1)} - e^{(k)} \cong \lambda_{\max}\left(e^{(k)} - e^{(k-1)}\right) \tag{2.128}$$

In an iterative methods, the solution vector $\boldsymbol{x}^{(k)}$ is modified in each iteration. Let's denote the modification vector as $\boldsymbol{d}^{(k)}$ and define as follows:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{d}^{(k)} \tag{2.129}$$

Using the definition of error vectors from eq (2.114) into (2.128) and combining with the relation defined by (2.129), we obtain,

$$d^{(k)} \cong \lambda_{\max} d^{(k-1)} \tag{2.130}$$

Now, recall equation (2.110) of Gauss-Seidel iteration and rewrite as,

$$x_i^{(k+1)} = x_i^{(k)} + \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1,\ 2,\ \cdots\ n \tag{2.131}$$

$$\text{or } x_i^{(k+1)} = x_i^{(k)} + d_i^{(k)}, \quad i = 1,\ 2,\ \cdots\ n \tag{2.132}$$

This is same as eq. (2.129). This means that, at every iteration, the solution vector is modified by a displacement vector $d$. Furthermore, implication of eq. (2.130) is that, for a $\lambda_{\max} > 0$, the

direction of the displacement vector does not change from iteration ($k$-1) to iteration ($k$) for large enough value of $k$. It is only the magnitude of displacement that changes and the magnitude is approximately given by the largest eigenvalue of the iteration matrix. This observation suggests relaxation procedure as follows:

$$x_i^{(k+1)} = x_i^{(k)} + \omega d_i^{(k)}, \quad i = 1,\ 2,\ \cdots\ n, \quad \omega > 0 \tag{2.133}$$

where, $\omega$ is known as a relaxation parameter. Depending on the value of $\omega$, the following relaxation procedures can be defined:

$0 < \omega < 1$ : Under relaxation

$\omega = 1$ : Gauss Seidel

$1 < \omega < 2$ : Over Relaxation

Using (2.133) in conjunction with (2.131), a modified Gauss Seidel iteration scheme can be written as follows:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}}, \quad i = 1,\ 2,\ \cdots\ n \tag{2.134}$$

When over relaxation is done in succession, the method is commonly known as method of *successive over relaxation (SOR)*.

**Example 2.13**: Solve the Batman's problem of (2.2) using Successive Over Relaxation (SOR) with $\omega$ as 1.1, 1.4, and 1.6. Compare the number of iterations required for computing the solution with precision of 0.001 or less.

**Solution**: We assume an initial solution vector as, $\begin{bmatrix} S_0 \\ u \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ and perform SOR iterations using equation (2.134). We tabulate below the computations for three values of $\omega$, 1.1, 1.4 and 1.6.

**$\omega$ = 1.1**

| Iteration No. | $S_0$ | $u$ | $f$ | $\varepsilon^{(k)}$ |
|---|---|---|---|---|
| | 0 | 0 | 0 | |
| 1 | 482,9 | 12,16971831 | 0,523948265 | 482,9 |
| 2 | 340,5352069 | 17,88896915 | 0,534049181 | 142,3647931 |
| 3 | 315,5528419 | 19,17340953 | 0,51642553 | 24,98236495 |
| 4 | 309,6637942 | 19,63880005 | 0,498300819 | 5,889047728 |
| 5 | 307,4619283 | 19,90438206 | 0,483683569 | 2,201865869 |
| 6 | 306,1798835 | 20,09129792 | 0,472408698 | 1,282044872 |
| 7 | 305,2716952 | 20,2310157 | 0,463795915 | 0,908188321 |
| 8 | 304,5917302 | 20,33698309 | 0,457231032 | 0,679964961 |
| 9 | 304,0758238 | 20,41762267 | 0,452229589 | 0,515906405 |
| 10 | 303,6831931 | 20,47903504 | 0,448419676 | 0,392630719 |
| 11 | 303,3841729 | 20,52581276 | 0,445517501 | 0,299020154 |
| 12 | 303,1564087 | 20,56144468 | 0,443306802 | 0,227764216 |

| | | | | |
|---|---|---|---|---|
| 13 | 302,982914 | 20,58858677 | 0,441622829 | 0,173494683 |
| 14 | 302,8507569 | 20,60926191 | 0,440340084 | 0,13215707 |
| 15 | 302,750088 | 20,62501093 | 0,439362969 | 0,100668927 |
| 16 | 302,6734047 | 20,63700755 | 0,438618665 | 0,076683277 |
| 17 | 302,6149922 | 20,64614582 | 0,4380517 | 0,058412518 |
| 18 | 302,5704972 | 20,65310679 | 0,437619821 | 0,044494999 |
| 19 | 302,5366037 | 20,65840921 | 0,437290843 | 0,033893504 |
| 20 | 302,5107858 | 20,66244827 | 0,437040249 | 0,02581795 |
| 21 | 302,4911193 | 20,66552497 | 0,436849361 | 0,019666498 |
| 22 | 302,4761386 | 20,66786861 | 0,436703955 | 0,014980707 |
| 23 | 302,4647272 | 20,66965385 | 0,436593194 | 0,011411364 |
| 24 | 302,4560347 | 20,67101373 | 0,436508823 | 0,008692463 |
| 25 | 302,4494134 | 20,6720496 | 0,436444554 | 0,006621374 |
| 26 | 302,4443696 | 20,67283866 | 0,436395599 | 0,005043748 |
| 27 | 302,4405276 | 20,67343972 | 0,436358307 | 0,003842012 |
| 28 | 302,437601 | 20,67389757 | 0,436329901 | 0,002926604 |
| 29 | 302,4353717 | 20,67424633 | 0,436308263 | 0,002229304 |
| 30 | 302,4336736 | 20,674512 | 0,43629178 | 0,001698144 |
| 31 | 302,43238 | 20,67471436 | 0,436279225 | 0,00129354 |
| 32 | 302,4313947 | 20,67486851 | 0,436269661 | 0,000985338 |

**$\omega$ = 1.4**

| **Iteration No.** | $S_0$ | $u$ | $f$ | $\varepsilon^{(k)}$ |
|---|---|---|---|---|
| | 0 | 0 | 0 | |
| 1 | 614,6 | 2,504225352 | 1,140096831 | 614,6 |
| 2 | 316,3455984 | 19,57533625 | 0,214535317 | 298,2544016 |
| 3 | 312,3751257 | 22,3384284 | 0,26153006 | 3,97047272 |
| 4 | 288,7151404 | 23,0987384 | 0,27972458 | 23,6599853 |
| 5 | 291,0900655 | 22,37961332 | 0,349084477 | 2,374925134 |
| 6 | 294,5157651 | 21,64008095 | 0,394665437 | 3,42569956 |
| 7 | 298,3381337 | 21,10596591 | 0,421867251 | 3,822368647 |
| 8 | 300,7133584 | 20,81504881 | 0,434096561 | 2,37522471 |
| 9 | 301,9593627 | 20,68701081 | 0,438256437 | 1,246004283 |
| 10 | 302,4603968 | 20,64747911 | 0,438738 | 0,5010341 |
| 11 | 302,5901604 | 20,64571146 | 0,438042204 | 0,129763622 |
| 12 | 302,5723207 | 20,65509359 | 0,437247309 | 0,017839755 |
| 13 | 302,5194088 | 20,66445866 | 0,436689901 | 0,052911896 |
| 14 | 302,4742835 | 20,67070224 | 0,436384231 | 0,045125299 |
| 15 | 302,4463642 | 20,67399577 | 0,436250608 | 0,027919254 |
| 16 | 302,4325397 | 20,67536956 | 0,436209479 | 0,013824551 |
| 17 | 302,4272517 | 20,67575021 | 0,436207858 | 0,005287986 |
| 18 | 302,4261065 | 20,67572697 | 0,436217802 | 0,001145238 |
| 19 | 302,4264987 | 20,67559876 | 0,436227662 | 0,000392209 |

**$\omega$ = 1.6**

| **Iteration No.** | $S_0$ | $u$ | $f$ | $\varepsilon^{(k)}$ |
|---|---|---|---|---|
| | 0 | 0 | 0 | |
| 1 | 702,4 | -7,030985915 | 2,105191865 | 702,4 |
| 2 | 285,96852 | 20,19448294 | -0,391462937 | 416,43148 |
| 3 | 342,5278855 | 25,84831394 | -0,061870437 | 56,55936544 |
| 4 | 242,3706341 | 29,99716864 | -0,213561485 | 100,1572514 |
| 5 | 265,9731494 | 26,57162777 | 0,216289879 | 23,60251532 |
| 6 | 272,5742165 | 23,0000586 | 0,426504586 | 6,601067058 |
| 7 | 297,5790198 | 19,9375184 | 0,578412096 | 25,00480328 |
| 8 | 308,2850768 | 18,84305692 | 0,575340804 | 10,70605707 |
| 9 | 312,8129488 | 19,02444092 | 0,522394845 | 4,527871952 |

| | | | | |
|---|---|---|---|---|
| 10 | 309,9250905 | 19,84246911 | 0,455718715 | 2,8878583 |
| 11 | 305,5933902 | 20,59717194 | 0,415535342 | 4,33170027 |
| 12 | 301,9414774 | 21,01231649 | 0,403629262 | 3,651912832 |
| 13 | 300,3805269 | 21,07436457 | 0,411863939 | 1,560950444 |
| 14 | 300,4483474 | 20,93594804 | 0,426264394 | 0,13841653 |
| 15 | 301,3379043 | 20,75517703 | 0,437775357 | 0,889556896 |
| 16 | 302,2434335 | 20,63084364 | 0,442855634 | 0,905529178 |
| 17 | 302,7772746 | 20,58758063 | 0,442583299 | 0,533841049 |
| 18 | 302,8945138 | 20,6034221 | 0,439735934 | 0,117239254 |
| 19 | 302,7545851 | 20,6420299 | 0,436821355 | 0,139928723 |
| 20 | 302,545182 | 20,67556955 | 0,435114203 | 0,209403079 |
| 21 | 302,3906089 | 20,6922557 | 0,434740556 | 0,154573133 |
| 22 | 302,3293165 | 20,69339481 | 0,435192909 | 0,061292346 |
| 23 | 302,3408812 | 20,68626956 | 0,435865636 | 0,01156467 |
| 24 | 302,3839372 | 20,67805115 | 0,436361467 | 0,043055993 |
| 25 | 302,4243825 | 20,67279235 | 0,436557284 | 0,040445273 |
| 26 | 302,4462608 | 20,67125797 | 0,436519459 | 0,021878379 |
| 27 | 302,449518 | 20,67224129 | 0,436380909 | 0,003257136 |
| 28 | 302,44207 | 20,67406445 | 0,436251087 | 0,007448019 |
| 29 | 302,4324453 | 20,67552979 | 0,436181027 | 0,009624704 |
| 30 | 302,4258383 | 20,67619091 | 0,436170754 | 0,006606914 |
| 31 | 302,4235601 | 20,67616765 | 0,436194771 | 0,002278219 |
| 32 | 302,4244193 | 20,67581197 | 0,436225693 | 0,000859146 |
| 33 | 302,4264812 | 20,67544177 | 0,436246875 | 0,002061944 |
| 34 | 302,428265 | 20,67522227 | 0,436254146 | 0,001783799 |
| 35 | 302,4291486 | 20,67517181 | 0,436251357 | 0,000883544 |

From the 3 tables above, it appears that there is an optimum value of $\omega$ for this problem. We plot the number of computations required to attain the same accuracy with different $\omega$ values.

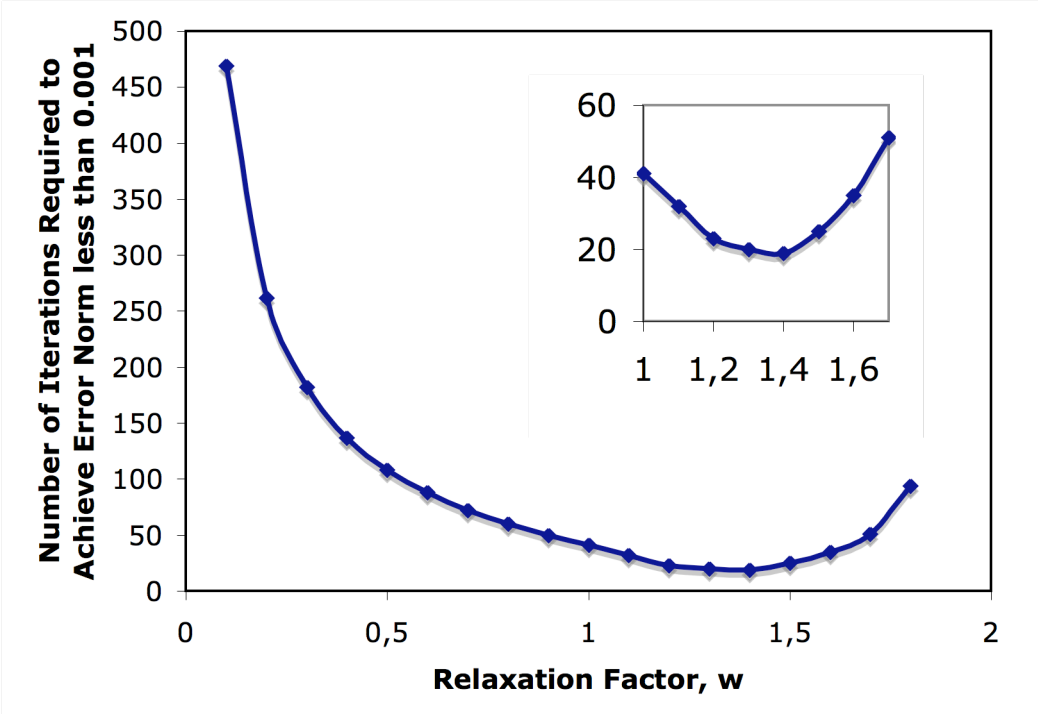



**Figure 2.6:** Number of Iterations Required vs. Relaxation Factor. Inset shows the optimum near 1.4.

Beyond the value of 1.8 for $\omega$, the solution starts to oscillate. In general, for a problem where Gauss Seidel oscillates (but converges), under relaxation will help to stabilize the oscillation. Over relaxation almost always speeds up the convergence for the problems where Gauss Seidel has a steady convergence. Number of iteration required was half with $\omega = 1.4$ compared to Gauss Seidel method for the same problem (Example 2.12). There is always an optimum value of the relaxation factor beyond which the convergence slows down and eventually leads to oscillation. The optimum value depends on the specific matrix.

### 2.2.3 *Scaling and Equilibration*

In a linear system of equation of engineering interest, unknowns are physical quantities. These can be expressed in various units. If the units of the variables are unmatched, we may have a coefficient matrix with values differing in orders of magnitude. The same may happen if one has done measurements with widely different parameters. For example, let us consider the following problem:

$$\begin{bmatrix} 0.003 & 1.45 & 0.3 \\ 0.00002 & 0.0096 & 0.0021 \\ 0.0015 & 0.966 & 0.201 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 0.12 \\ 19 \end{bmatrix}$$

It is evident by looking at the coefficient matrix that the first column elements are about 3 orders of magnitude smaller compared to the other elements. These elements correspond to the variable $x_1$. This may sometimes appear because of a different unit in $x_1$. For example, $x_1$ may have been entered in centimetre while the other variables were entered in metres. The second row elements are about 2 orders of magnitude smaller compared to the other elements. This may have happened due to range of measurement corresponding to second equation being much smaller compared to the other two equations. If one wants to solve this equation by the method of Gauss Elimination, after 1st elimination step, we obtain the following coefficient matrix:

$$\begin{bmatrix} 0.003 & 1.45 & 0.3 \\ 0 & 0.0000667 & 0.0001 \\ 0 & 0.241 & 0.041 \end{bmatrix}$$

If one is doing the calculation by 4 digits, the 2nd pivot will be approximated as 0.0001. Carrying on, one finds the solution vector as [-22328.9, -35.49, 431.5]. The True solution vector is [-22400, -37.454545, 441.69697]. However, if we *scale* the variable $x_1$ as $x_1 = 10^3 \times x_1'$ and replace $x_1$ by $x_1'$ the set of equation becomes:

$$\begin{bmatrix} 3 & 1.45 & 0.3 \\ 0.02 & 0.0096 & 0.0021 \\ 1.5 & 0.966 & 0.201 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 0.12 \\ 19 \end{bmatrix}$$

Now, we can multiply the second equation by 100 in order to bring the values in the coefficient matrix to the same order of magnitude as the rest of the values. This operation is called *equilibration*. Following the equilibration step, the equation becomes:

$$\begin{bmatrix} 3 & 1.45 & 0.3 \\ 2 & 0.96 & 0.21 \\ 1.5 & 0.966 & 0.201 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ 19 \end{bmatrix}$$

Now, if one solves the set of equation with 4 digit arithmetic, one obtains the solution vector as [-22.456, -37.336, 441.685]. To get the original solution, we have to multiply the first element with the scaling factor. We observe that the solution improved. Essentially, we reduced round-off error by making the order of magnitudes of the elements much closer compared to the original form. Recall that round-off error increases when we take small differences of two very small or very large numbers (Chapter 1).

For incorporation of these operations in matrix form into an algorithm, let us make some observations. We will consider the original set of equations of the form $Ax = b$.

Scaling operation essentially involves multiplying each column by a constant factor. Some columns may have this factor as one and some may have a scale. However, this essentially translates into multiplying the scaled variable with a diagonal scaling matrix to yield the original variable, i.e., $x = Sx'$. For the above example, it was,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10^3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2 \\ x_3 \end{bmatrix}$$

Notice, that scale factors of corresponding elements are along the diagonals of matrix $S$. Using this relation, the original set of equation becomes $Ax = ASx' = b$. The modified equation gives the solution vector for $x'$ and the original vector can be obtained by using the relation $x = Sx'$. Notice that the new coefficient matrix is $A$ post multiplied by the diagonal matrix $S$. This yields the required effect of multiplying the columns with related scale factor. For our example:

$$\begin{bmatrix} 0.003 & 1.45 & 0.3 \\ 0.00002 & 0.0096 & 0.0021 \\ 0.0015 & 0.966 & 0.201 \end{bmatrix} \begin{bmatrix} 10^3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1.45 & 0.3 \\ 0.02 & 0.0096 & 0.0021 \\ 1.5 & 0.966 & 0.201 \end{bmatrix}$$

Similarly, equilibration is premultiplying the coefficient matrix with a diagonal matrix with equilibration factors along the diagonal. So, the equation after scaling and equilibration would become,

$$EASx' = Eb \tag{2.135}$$

where, $E$ and $S$ are diagonal matrices containing the equilibration and scaling factors on the respective diagonals.

Notice that the right hand side vector also changed. We have seen this for our example problem. So, for our example problem, final coefficient matrix was:

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 10^2 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0.003 & 1.45 & 0.3\\ 0.00002 & 0.0096 & 0.0021\\ 0.0015 & 0.966 & 0.201\end{bmatrix}\begin{bmatrix}10^3 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}3 & 1.45 & 0.3\\ 2 & 0.96 & 0.21\\ 1.5 & 0.966 & 0.201\end{bmatrix}$$

Final right hand side vector was:

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 10^2 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}11\\ 0.12\\ 19\end{bmatrix} = \begin{bmatrix}11\\ 12\\ 19\end{bmatrix}$$

We see that we need to decide on scaling factors and equilibration factors before we start our solution procedure. This is true for both direct and iterative methods. Once, decided, it can be easily implemented by incorporating a few matrix multiplication steps.

Scaling and equlibration do not eliminate the need for pivoting operation. You will notice in the example that after step 1, the pivot becomes small. Exchanging row 2 and 3 in a partial pivoting operation eliminates division by a small number. This is regardless of whether scaling and/or equlibration was done or not. Moreover, while scaling and equilibration helps in reducing round-off errors, it does not transform an *ill-conditioned* matrix into a *well-conditioned* one.

## Exercise 2.2

1. Solve the following system of equations by Gauss Elimination, Doolittle's method, Crout's method and Cholesky decomposition:

$$\begin{bmatrix}9.3746 & 3.0416 & -2.4371\\ 3.0416 & 6.1832 & 1.2163\\ -2.4371 & 1.2163 & 8.4429\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} = \begin{bmatrix}9.2333\\ 8.2049\\ 3.9339\end{bmatrix}$$

2. Solve the following system of equation using Thomas algorithm:

$$\begin{bmatrix}-2 & 1 & 0 & 0\\ 1 & -4 & 1 & 0\\ 0 & 1 & -4 & 1\\ 0 & 0 & 1 & -2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} = \begin{bmatrix}3\\ 1\\ 2\\ -2\end{bmatrix}$$

3. Two approximate solutions to the following set of equations are given as described below:

$$\begin{bmatrix}10 & 7 & 8 & 7\\ 7 & 5 & 6 & 5\\ 8 & 6 & 10 & 9\\ 7 & 5 & 9 & 10\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} = \begin{bmatrix}32\\ 23\\ 33\\ 31\end{bmatrix}$$

An approximation to the x-values as [-7.2, 14.6, -2.5, 3.1] yields the right hand side vector as [31.9, 23.1, 32.9, 31.1]. A very different set of x-values [0.18, 2.36, 0.65, 1.21] also yields a very close right hand side vector as [31.99, 23.01, 32.99, 31.01]. It is not clear whether any of the x-values are close to the true solution. Use Crout's decomposition and improve the solution starting from each of the above approximations of x-values.

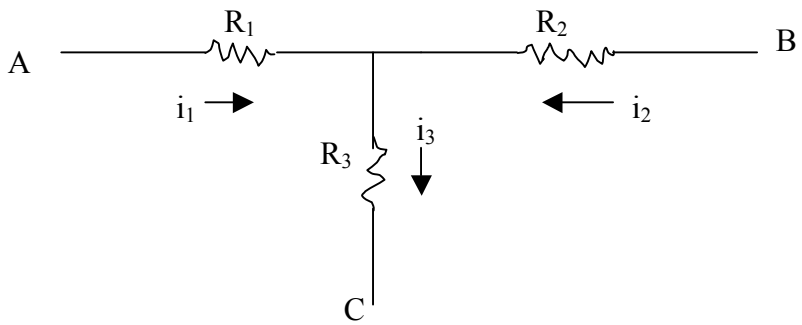
4. Consider the following set of equations:

$$\begin{bmatrix} 10^{-5} & 10^{-5} & 1 \\ 10^{-5} & 10^{-5} & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2\times10^{-5} \\ -2\times10^{-5} \\ 1 \end{bmatrix}$$

a) Solve the system using Gaussian elimination, without pivoting, using 3-digit floating-point arithmetic with round-off.
b) Perform complete pivoting and carry out Gaussian elimination steps once again using 3-digit floating-point arithmetic with round-off. Explain the results.

c) Rewrite the set of equations after scaling according to $x'_3 = 10^5 \times x_3$ and equilibration.

5. For the circuit shown in the figure, find the currents through the elements using (a) Gauss elimination (b) Jacobi Iteration, (c) Gauss Seidel method, (d) SOR method. Compare the number of iterations taken by methods (b), (c) and (d). (Use $R_1$=10 Ohm; $R_2$=20 Ohm; $R_3$=40 Ohm; $V_A$-$V_C$ = 200 Volt; $V_B$-$V_C$ = 100 Volt)




6. In the above problem, how much will be the change in current in $R_3$ due to unit change in voltage difference ($V_A$-$V_C$)? Which of the currents is the most sensitive to a change in voltage at B?

7. Solve the following set of equations using (a) Doolittle and Crout decomposition (b) Thomas algorithm and (c) Cholesky decomposition

$$4x_1 + x_2 = 6$$
$$x_1 + 4x_2 + x_3 = 12$$
$$x_2 + 4x_3 = 14$$

8. Consider the following matrix:

$$A = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix}$$

a) Compute $A^{-1}$ using Cholesky decomposition.

b) Find the solution of $Ax=b$ where, $b = [4\ 3\ 3\ 1]^T$.

c) If $b$ is perturbed by $\delta b$ such that $\| \delta b \|_\infty = 0.01$, find an upper bound for the corresponding perturbation vector $\| \delta x \|_\infty$.

d) Compute the condition number $\ell(A)$ and check the inequality $\dfrac{\|\delta x\|_\infty}{\|x\|_\infty} \Big/ \dfrac{\|\delta b\|_\infty}{\|b\|_\infty} \le \ell(A)$.

9. The following system of five equations can be solved using Thomas Algorithm for tri-diagonal matrices:

$$\begin{array}{cccc:cc:c} & \boldsymbol{A} & & & \boldsymbol{b} & \boldsymbol{x} & \boldsymbol{d} \\ \begin{bmatrix} 4 & 1 & 0 & 0 \\ -2 & 10 & 1 & 0 \\ 0 & -3 & 8 & 1 \\ 0 & 0 & 1 & 5 \end{bmatrix} & & & & \begin{bmatrix} -2 \\ -1 \\ -2 \\ 3 \end{bmatrix} & \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} & = \begin{bmatrix} 4.75 \\ -1.15 \\ 0.55 \\ 12.35 \end{bmatrix} \\ \hdashline \begin{bmatrix} -3 & 2 & 1 & -5 \end{bmatrix} & & & & 13 & x_5 & = -4.15 \\ \boldsymbol{c^T} & & & & a_{55} & x_5 & d_5 \end{array}$$

The system can be partitioned as shown above and expressed as $\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{c}^T & a_{55} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ x_5 \end{bmatrix} = \begin{bmatrix} \boldsymbol{d} \\ d_5 \end{bmatrix}$.

Answer the following:

a) If vector $\boldsymbol{x}$ is decomposed into two parts as $\boldsymbol{x} = \boldsymbol{x}^H + \boldsymbol{x}^I$ such that, vector $\boldsymbol{x}^H$ is the solution of tri-diagonal system $\boldsymbol{A}\boldsymbol{x}^H = \boldsymbol{d}$, prove the following:

$$\boldsymbol{x}^I = -x_5 \boldsymbol{y}$$

$$x_5 = \frac{d_5 - \boldsymbol{c}^T \boldsymbol{x}^H}{a_{55} - \boldsymbol{c}^T \boldsymbol{y}}$$

where, the vector $\boldsymbol{y}$ is the solution of tri-diagonal system of equations and $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$.

b) Using the results of (a), compute the solution consisting of vector $\boldsymbol{x}$ and variable $x_5$.

10. Consider the following matrix:

$$A = \begin{bmatrix} 9 & 3 & -2 \\ 3 & 6 & 1 \\ -2 & 1 & 9 \end{bmatrix}$$

a) Reduce it to an upper triangular matrix using Gauss Elimination Procedure.

b) Synthesize a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$ from the steps of (a) above such that $\boldsymbol{A} = \boldsymbol{LU}$.

c) Compute $\boldsymbol{A}^{-1}$ using the $\boldsymbol{LU}$ decomposition obtained in (b).

d) Using the above results, compute the determinant and condition number of $\boldsymbol{A}$. Use row sum norms for the matrices.

11. Consider the family of so-called Hilbert matrices, given by ($n$ = 1,2,….):

$$H_n = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & \ldots & 1/n \\ 1/2 & 1/3 & 1/4 & 1/5 & \ldots & 1/(n+1) \\ 1/3 & 1/4 & 1/5 & 1/6 & \ldots & 1/(n+2) \\ 1/4 & 1/5 & 1/6 & 1/7 & \ldots & 1/(n+3) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1/n & 1/(n+1) & 1/(n+2) & 1/(n+3) & \ldots & 1/(2n-1) \end{bmatrix}$$

a) Solve the system of equations $H_3x = b$, where x is a column vector of 3 variables and $b^T = [11/6, 13/12, 47/60]$.

b) Write down the $LU$ decomposition of $H_3$ from Gaussian elimination steps.

c) What is the condition number of $H_3$ ?

12. Show that a square matrix of size $n$ with $n$ independent columns will have a decomposition of the form $A = LDU$.

13. Derive a sufficient condition for the convergence of Gauss-Seidel iteration method similar to equation (2.125) for Jacobi iteration.

## 2.3 Computation of Eigenvalues

Concepts of eigenvalues and eigenvectors as elements of invariant subspace of the linear transformation were introduced in section 2.1.2. At this point, we will recall that the eigenvalues of a matrix is the solution of the polynomial $\det(A-\lambda I) = 0$. The eigenvalues are important in many engineering problems. The problem also appears from a group of ordinary differential equation known as eigenvalue problems. Sometimes, it is sufficient to get the

range within which the eigenvalues lie. For example, to compute the condition number of a symmetric matrix, we only need its largest and smallest eigenvalues. In more precise engineering analysis, one requires precise estimation of all the eigenvalues. In this section, we will first discuss a simple iterative process known as Power Method to estimate the largest eigenvalue. A variation of this method can also be used to determine the smallest eigenvalue. Then we will present a method to determine all the eigenvalues from the first principle by solving the characteristic polynomial. At the end we will discuss a method of triangulation/diagonalization by which all the eigenvalues can be estimated simultaneously.

### *2.3.1 The Power Method*

If an $n \times n$ matrix $A$ has a unique eigenvalue of maximum absolute magnitude ($\lambda_1$) such that, $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots\ldots \geq |\lambda_n|$ and $n$ independent eigenvectors $\{x_1, x_2 \cdots\cdots x_n\}$, this method provides an easy way to estimate it. Starting from an arbitrary non-zero vector ($z^{(0)}$) of length $n$, the method generates a sequence of vectors ($z^{(k)}$) through an iterative process that converges to the eigenvector corresponding to the maximum eigenvalue. The iteration scheme is given by:

$$z^{(k+1)} = Az^{(k)} = A^{k+1}z^{(0)} \tag{2.136}$$

The proof that the vector $z^{(k)}$ will converge to the eigenvector corresponding to the maximum eigenvalue involves expressing the arbitrary initial vector $z^{(0)}$ as a linear combination of $n$ linearly independent eigenvectors as in equation (2.120) followed by application of the iterative scheme described above in equation (2.136).

$$z^{(0)} = \alpha_1 x_1 + \alpha_2 x_2 + \cdots\cdots + \alpha_n x_n \tag{2.137}$$

where, $x_i$ is the eigenvector corresponding to the eigenvalue $\lambda_i$ and $\alpha_i$'s are the constants. Since, by definition of eigenvalue $Ax_j = \lambda_j x_j$, therefore, it follows,

$$z^{(k)} = A^k z^{(0)} = A^k \sum_{i=1}^{n} \alpha_i x_i = \sum_{i=1}^{n} \lambda_i^k \alpha_i x_i = \lambda_1^k \left( \alpha_1 x_1 + \sum_{j=2}^{n} \left( \frac{\lambda_j}{\lambda_1} \right)^k \alpha_j x_j \right) \tag{2.138}$$

Since, $\lambda_1$ is a unique maximum eigenvalue satisfying $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots\ldots \geq |\lambda_n|$, the ratio $\left( \frac{\lambda_j}{\lambda_1} \right)$ approaches zero for large $k$. This essentially means that the vector $z^{(k)}$ approaches the same direction as $x_1$. Since, the magnitude of the maximum eigenvalue $\lambda_1$ are often different from unity, the elements of $z^{(k)}$ may approach zero (for $\lambda_1 < 1$) or become very large (for $\lambda_1 > 1$) with large $k$ because of the factor $\lambda_1^k$. Therefore, it is a good practice to normalize the vector $z^{(k)}$ after every iteration with its norm. Any suitable vector norm can be used for this purpose. The $L_2$ norm is shown below for illustration of the final iteration scheme:

$$\begin{aligned} y^{(k)} &= \frac{z^{(k)}}{\left\| z^{(k)} \right\|_2} \\ z^{(k+1)} &= Ay^{(k)} \end{aligned} \tag{2.139}$$

Termination criteria will be when $y^{(k)}$ becomes stationary or the variation is within a chosen tolerance. Once an eigenvector ($x_i$) is known, the corresponding eigenvalue ($\lambda_i$) can be estimated using the *Rayleigh quotient* given by $\lambda_i = \dfrac{x_i^T A x_i}{x_i^T x_i}$. An approximation of $\lambda_1$ at $k^{\text{th}}$ iteration can therefore be obtained as follows:

$$\lambda_1^{(k)} = \frac{y^{(k)T} A y^{(k)}}{y^{(k)T} y^{(k)}} = y^{(k)T} z^{(k+1)} \text{ since, } y^{(k)T} y^{(k)} = 1 \text{ and } z^{(k+1)} = A y^{(k)}. \tag{2.140}$$

Termination criterion can either be applied either on the eigenvalue $\lambda_1^{(k)}$ or on the eigenvector $y^{(k)}$. We can set a tolerance for either absolute approximate error or relative approximate error. If applied on eigenvalue, one can work with absolute value of the difference between successive iterations. For the eigenvector, one will have to choose a vector norm. Any norm described in Chapter 1 can be used. However, Euclidean or the maximum norm are mostly used. The error expressions are shown below:

For Eigenvalue: $\varepsilon^{(k)} = \left|\lambda_1^{(k+1)} - \lambda_1^{(k)}\right|$ or $\varepsilon_r^{(k)} = \left|\dfrac{\lambda_1^{(k+1)} - \lambda_1^{(k)}}{\lambda_1^{(k+1)}}\right|$

For Eigenvectors: $\varepsilon^{(k)} = \left\|y^{(k+1)} - y^{(k)}\right\|$ or $\varepsilon_r^{(k)} = \dfrac{\left\|y^{(k+1)} - y^{(k)}\right\|}{\left\|y^{(k+1)}\right\|}$

**Example 2.14:** Compute the largest eigenvalue and the corresponding eigenvector of the following coefficient matrix of Example 2.9 using the power method with a relative precision of 0.1% or less on the eigenvalue.

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

**Solution:** We assume an initial vector $z_0=[1\ 0\ 0\ 0]^T$. We now apply the algorithm given by equation (2.139) and (2.140). The computations are tabulated below:

| Iteration No. | *Y* | *z* | $\lambda$ | $\varepsilon$ (%) |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | |
| | 0 | -1 | | |
| | 0 | 0 | | |
| | 0 | 0 | | |
| 2 | 0.894427 | 2.236068 | 3.2 | 60 |
| | -0.44721 | -2.68328 | | |
| | 0 | 0.447214 | | |
| | 0 | 0 | | |
| 3 | 0.635001 | 2.032002 | 4.354839 | 36.08871 |
| | -0.762 | -3.81 | | |
| | 0.127 | 1.270001 | | |
| | 0 | -0.127 | | |
| 4 | 0.451287 | 1.748735 | 4.848051 | 11.32561 |

| | | | | |
|---|---|---|---|---|
| | -0.84616 | -4.11799 | | |
| | 0.282054 | 2.002584 | | |
| | -0.02821 | -0.33846 | | |
| 5 | 0.355916 | 1.549959 | 5.073126 | 4.642583 |
| | -0.83813 | -4.116 | | |
| | 0.407582 | 2.53734 | | |
| | -0.06889 | -0.54536 | | |
| 6 | 0.303509 | 1.413004 | 5.18924 | 2.288802 |
| | -0.80599 | -4.02431 | | |
| | 0.496856 | 2.900198 | | |
| | -0.10679 | -0.71044 | | |
| 7 | 0.271393 | 1.315727 | 5.248124 | 1.134739 |
| | -0.77294 | -3.92019 | | |
| | 0.557036 | 3.137536 | | |
| | -0.13645 | -0.82994 | | |
| 8 | 0.250299 | 1.246363 | 5.276903 | 0.548367 |
| | -0.74576 | -3.83023 | | |
| | 0.596874 | 3.291145 | | |
| | -0.15788 | -0.91264 | | |
| 9 | 0.23601 | 1.19731 | 5.290633 | 0.260201 |
| | -0.72529 | -3.76038 | | |
| | 0.623209 | 3.390941 | | |
| | -0.17282 | -0.96884 | | |
| 10 | 0.226225 | 1.162954 | 5.297102 | 0.122255 |
| | -0.7105 | -3.70894 | | |
| | 0.6407 | 3.456363 | | |
| | -0.18306 | -1.00682 | | |
| 11 | 0.219508 | 1.13908 | 5.300129 | 0.057163 |
| | -0.70006 | -3.67215 | | |
| | 0.65239 | 3.499661 | | |
| | -0.19004 | -1.03246 | | |

The error has become less than the specified 0.1%. So, the iterations may be terminated. The maximum eigenvalue is therefore 5.300129 and the corresponding eigenvector is given by the final z or by normalizing we obtain $[0.214898 \;\; -0.69279 \;\; 0.660245 \;\; -0.19478]^T$.

### *2.3.2 The Inverse Power Method*

If a $n \times n$ matrix $A$ has an unique eigenvalue of minimum absolute magnitude, it can also be obtained using the same principle as the power method described above. If $\lambda_i$'s are the eigenvalues of matrix $A$, the inverse of the matrix have the eigenvalues as $\frac{1}{\lambda_i}$. Thus, the unique eigenvalue of minimum magnitude of the matrix $A$ will translate into a unique eigenvalue of maximum magnitude for the matrix $A^{-1}$. Thus, application of the power method on $A^{-1}$ will converge to the inverse of the minimum eigenvalue of matrix $A$. The algorithm can be written similar to equation (2.139) as follows:

$$y^{(k)} = \frac{z^{(k)}}{\left\| z^{(k)} \right\|_2}$$
$$z^{(k+1)} = A^{-1} y^{(k)} \tag{2.141}$$

where, $z^{(0)}$ is an arbitrary non-zero vector of length $n$ (assumed). One can avoid inversion of matrix $A$. Instead, an $LU$ decomposition can be carried out for $A$ which can then be used

repeatedly to solve for $z^{(k+1)}$ with known right hand side vector $y^{(k)}$. This gives a significant computational advantage. Thus, the above algorithm translates to,

$$y^{(k)} = \frac{z^{(k)}}{\left\| z^{(k)} \right\|_2}$$
$$Az^{(k+1)} = y^{(k)} \tag{2.142}$$

Once, the $y^{(k)}$ converges, one can calculate the corresponding eigenvalue using the Rayleigh quotient described in the previous section. Do not forget that it is the maximum eigenvalue of $A^{-1}$. In order to get the minimum eigenvalue of $A$, you have to invert it.

**Example 2.15:** Compute the smallest eigenvalue of the matrix in Example 2.14 with a relative precision of 0.3% or less.

**Solution:** We use the computation logic given by equation (2.142). So, at every step, we solve for $z_{k+1}$ for a known $y_k$. At this point we observe that the matrix in Example 2.14 is a tridiagonal. Thus, the Thomas Algorithm described in section 2.2.1.4. Since, the coefficient matrix remain unaltered, the vector $\alpha$ can be calculated *apriori* using eq (2.86), before the start of the power method iterations. At every step of the power method, right hand side vector $\beta$ needs to be recalculated using (2.86) and then the solutions can be easily computed using (2.88).

Since, the matrix of example 2.9 is the same as the present problem, we use the values of vector $\alpha$ computed there as $\alpha = \begin{bmatrix} 2 & 3.5 & 3.714286 & 1.730769 \end{bmatrix}^T$. We assume the same initial vector as in example 2.14 and tabulate the iterations of Inverse Power Method.

| Iteration No. | $y$ | $\beta$ | $z$ | $\lambda$ | $\varepsilon$ (%) |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0.571429 | 0.571429 | |
| | 0 | -0.5 | -0.14286 | | |
| | 0 | 0 | 0 | | |
| | 0 | 0 | 0 | | |
| 2 | 0.970143 | 0.970143 | 0.59168 | 0.625727 | 8.677686 |
| | -0.24254 | -0.72761 | -0.21322 | | |
| | 0 | 0.069296 | 0.018657 | | |
| | 0 | 0 | 0 | | |
| 3 | 0.940366 | 0.940366 | 0.590804 | 0.638367 | 1.980087 |
| | -0.33887 | -0.80905 | -0.24124 | | |
| | 0.029651 | 0.126471 | 0.035292 | | |
| | 0 | -0.00798 | -0.00461 | | |
| 4 | 0.924358 | 0.924358 | 0.588887 | 0.642697 | 0.673696 |
| | -0.37744 | -0.83962 | -0.25342 | | |
| | 0.055217 | 0.163057 | 0.047335 | | |
| | -0.00722 | -0.02208 | -0.01276 | | |
| 5 | 0.915885 | 0.915885 | 0.587712 | 0.645169 | 0.383143 |
| | -0.39413 | -0.85208 | -0.25954 | | |
| | 0.073619 | 0.186229 | 0.056308 | | |
| | -0.01984 | -0.03966 | -0.02292 | | |
| 6 | 0.910704 | 0.910704 | 0.586942 | 0.647136 | 0.303866 |
| | -0.40217 | -0.85753 | -0.26318 | | |
| | 0.087254 | 0.202161 | 0.063606 | | |
| | -0.03551 | -0.059 | -0.03409 | | |
| 7 | 0.906781 | 0.906781 | 0.586251 | 0.648932 | 0.276877 |

| | | | | | |
|---|---|---|---|---|---|
| | -0.40659 | -0.85998 | -0.26572 | | |
| | 0.098267 | 0.214436 | 0.070041 | | |
| | -0.05267 | -0.07912 | -0.04572 | | |

We stop the iteration as soon as the relative error in eigenvalue falls below the specified 0.3%. Thus, the minimum eigenvalue is given by (1/0.648932) = 1.540993.

It is important to note here that if the matrix is not a tridiagonal, it will be computationally beneficial to perform a *LU* decomposition of the coefficient matrix in the beginning. At every iteration, the solution can be easily computed using the new right hand side vector.

### *2.3.3 The Inverse Power Method with Shift*

By extending the above principles, one can easily determine an eigenvalue closest to any given number ($\theta$) for a matrix $A$. Eigenvalues of matrix ($A$-$\theta I$) are ($\lambda_i$-$\theta$). Thus, the smallest absolute value of ($\lambda_i$-$\theta$) will be given by the eigenvalue closest to $\theta$. If it is the $p^{th}$ (say) eigenvalue which is closest to $\theta$ in magnitude, ($\lambda_p$-$\theta$) is the minimum eigenvalue of absolute magnitude for the matrix ($A$-$\theta I$). Applying the inverse power method on matrix ($A$-$\theta I$), one can estimate ($\lambda_p$-$\theta$) and in turn $\lambda_p$, since $\theta$ is known. This is important in many engineering application, because one sometimes is interested in finding the nearest eigenvalue to a predetermined number. As the eigenvalue is related to some physical quantity, this predetermined value may come from some guideline or design code provision.

### *2.3.4 Faddeev Leverrier Method*

First method we are going to illustrate for estimating all the eigenvalues of a matrix is based on the definition of eigenvalues given in section 2.1.2, *i.e.,* root of polynomial det($A$-$\lambda I$) = 0. For a square matrix of size $n$, this will lead to a *characteristic polynomial* of order $n$. The polynomial equation can be written as:

$$\lambda^n - a_{n-1}\lambda^{n-1} - \cdots - a_2\lambda^2 - a_1\lambda - a_0 = 0 \tag{2.143}$$

The *Fadeev-Leverrier* method is essentially a way to evaluate the coefficients ($a_i$'s) of the *characteristic polynomial* without evaluating the determinant explicitly. We initialize a sequence of matrices as:

$A_{n-1} = A$. Then, the coefficient $a_{n-1}$ is given by $a_{n-1}$ = trace ($A_{n-1}$). (2.144)

The other matrices in the sequence and the coefficients in the characteristic polynomial are computed as follows:

$$A_i = A\left(A_{i+1} - a_{i+1}I\right) \text{ and } a_i = \frac{trace\left(A_i\right)}{n-i}, \text{ where } i = (n-2), (n-3), \cdots, 2, 1, 0\,. \tag{2.145}$$

Proof of the fact that the coefficients of the polynomial are indeed given by the traces of the matrices generated recursively (by 2.145) is beyond the scope of the book. There are essentially two approaches, the original Faddeev's approach and a later one using Laplace Transform. The interested readers may consult Faddeeva (1959) and, Faddeeva and Faddeev

(1977) for the detailed proof using the matrix properties. The proof based on Laplace transform is available in Hou (1998).

Once the characteristic polynomial is formulated, any numerical method for finding the roots of a polynomial such as *Bairstow method* described in section 3.4 may be used to compute the eigenvalues.

**Example 2.16:** Formulate the characteristic polynomial using Faddeev Leverrier method for the matrix in example 2.14.

**Solution:** For $n$=4, we can express the characteristic polynomial as $\left\{\lambda^4 - a_3\lambda^{31} - a_2\lambda^2 - a_1\lambda - a_0\right\} = 0$. Thus, $A_3 = A$, and $a_3$ = trace($A$) = 12. We tabulate the computation of other $a_i$'s below:

| $i$ | $(A_{i+1}-a_{i+1}I)$ | | | | $A_i=A(A_{i+1}-a_{i+1}I)$ | | | | $a_i$ |
|---|---|---|---|---|---|---|---|---|---|
| 2 | -10 | -1 | 0 | 0 | -19 | 6 | 1 | 0 | -49 |
| | -1 | -8 | -1 | 0 | 6 | -30 | 4 | 1 | |
| | 0 | -1 | -8 | -1 | 1 | 4 | -30 | 6 | |
| | 0 | 0 | -1 | -10 | 0 | 1 | 6 | -19 | |
| 1 | 30 | 6 | 1 | 0 | 54 | -7 | -2 | -1 | 80 |
| | 6 | 19 | 4 | 1 | -7 | 66 | -4 | -2 | |
| | 1 | 4 | 19 | 6 | -2 | -4 | 66 | -7 | |
| | 0 | 1 | 6 | 30 | -1 | -2 | -7 | 54 | |
| 0 | -26 | -7 | -2 | -1 | -45 | 0 | 0 | 0 | -45 |
| | -7 | -14 | -4 | -2 | 0 | -45 | 0 | 0 | |
| | -2 | -4 | -14 | -7 | 0 | 0 | -45 | 0 | |
| | -1 | -2 | -7 | -26 | 0 | 0 | 0 | -45 | |

Therefore, the characteristic polynomial is given by $\lambda^4 - 12\lambda^3 + 49\lambda^2 - 80\lambda + 45 = 0$.

## *2.3.5 Similarity Transformation*

In this section we will present a method that is frequently used to compute eigenvalues of a matrix. It is based on similarity transformation. A sequence of matrices is generated by repeated application of similarity transformations. The eigenvalues of the original matrix remain unchanged under similarity transformations. The original matrix in the first step and the transformed matrices thereafter are decomposed into an orthogonal matrix and an upper triangular matrix. Properties of orthogonal matrix help simplify the process of similarity transformation. Under repeated similarity transformation, the matrix transforms into a diagonal or an upper triangular matrix. Since, determinant of a diagonal or triangular matrix is the product of the diagonal elements, the diagonal elements are the eigenvalues.

But, before we can describe the method, we need some more discussion of matrix algebra. We will first present the concept of similar matrices and similarity transformation. We will proceed to establish the condition under which a matrix can be diagonalized in order to estimate the eigenvalues using this method. We will then present a method for decomposition of a matrix into an orthogonal and an upper triangular matrix. Finally, we will present the algorithme based on this decomposition for estimation of the eigenvalues. At the end, we will also present the theorem and an outline of its proof which guarantees that the method eventually converges to an upper triangular matrix.

Two $n \times n$ matrices $A$ and $B$ are similar if there exists another $n \times n$ invertible matrix $S$ such that $A = SBS^{-1}$. In terms of linear transformation, their relationship is given by theorem 2.5.

**Theorem 2.5:** Two matrices $A$ and $B$ are similar if and only if they represent the same linear transformation $T : X \mapsto X$ with respect to two different bases.

**Proof:** Let $A$ represent $T$ with respect to the basis $\{x_1, x_2 \cdots x_n\}$ and $B$ represent $T$ with respect to the basis $\{y_1, y_2 \cdots y_n\}$, an arbitrary vector $x$ in $X$ and its transformation $Tx$ can be expressed as a linear combination of both the bases as follows:

$$x = \sum_i \alpha_i x_i \Rightarrow Tx = \sum_i (A\alpha)_i x_i \tag{2.146}$$

$$x = \sum_j \beta_j y_j \Rightarrow Tx = \sum_j (B\beta)_j y_j \tag{2.147}$$

Since, both $\{x_1, x_2 \cdots x_n\}$ and $\{y_1, y_2 \cdots y_n\}$ are bases for subspace $X$, each $y_j$ can also be expressed as a unique linear combination of $\{x_1, x_2 \cdots x_n\}$ as follows:

$$y_j = \sum_i s_{ij} x_i \quad \text{for} \quad j = 1,2,\cdots n \tag{2.148}$$

Now for an arbitrary vector $x = \sum_j \beta_j y_j$ in $X$, we may write the following:

$$x = \sum_j \beta_j y_j \Rightarrow Tx = \sum_j (B\beta)_j y_j = \sum_j (B\beta)_j \sum_i s_{ij} x_i = \sum_i (SB\beta)_i x_i \tag{2.149}$$

$$x = \sum_j \beta_j y_j = \sum_j \beta_j \sum_i s_{ij} x_i = \sum_i (S\beta)_i x_i \Rightarrow Tx = \sum_i (AS\beta)_i x_i \tag{2.150}$$

This gives $AS\beta = SB\beta$. Since $x$ was chosen arbitrarily, the relationship is valid for all $\beta$ in $\Re^n$. So, $AS = SB$.

We have already commented on the uniqueness of $s_{ij}$'s. Since, $y_j$'s are linearly independent, so will be the columns of $S$. This means the matrix $S$ is invertible. This leads to the required relation $A = SBS^{-1}$.

If you follow the above logic, the "only if" part can be easily shown by assuming that both $A$ and $B$ represent the transformation $T$ with the same basis vector and arriving at a contradiction.

Notice that in order to generate a similar matrix $B$ from original matrix $A$, we need to apply the following transformation: $B = S^{-1}AS$. The operation $S^{-1}AS$ will be termed as *similarity transformation* of $A$. Eigenvalues of two similar matrices are same. This can be easily seen from the definition of similar matrices as, $(A-\lambda I) = S(B-\lambda I)S^{-1}$. Since, determinants follow $\det(AB) = \det(A).\det(B)$, we find that $A$ and $B$ have same characteristic equations.

Some matrices can be diagonalized by similarity transform. That is to say, matrix $A$ can be diagonalized if it is similar to a diagonal matrix, i.e., $A = X\Lambda X^{-1}$. Then the eigenvalues of $A$ will be the diagonal elements of $\Lambda$. Since, this also means $AX = X\Lambda$, every column of $X$ is an

eigenvector $A$ and together, they span the invariant subspace of $A$. All non-defective matrices defined in section 2.1.2 are diagonalizable. Specifically, one will get more insight into diagonalizability by taking a closer look at the following theorems 2.6 and 2.7.

**Theorem 2.6:** If a $n \times n$ square matrix $A$ has $n$ linearly independent eigenvectors as $\{x_1, x_2 \cdots\cdots x_n\}$ then $A = X\Lambda X^{-1}$ where, $\Lambda$ is a diagonal matrix with $\{\lambda_1, \lambda_2 \cdots\cdots \lambda_n\}$ on its diagonal and $X$ is an $n \times n$ square matrix with eigenvectors $\{x_1, x_2 \cdots\cdots x_n\}$ on its columns.

**Proof:** By definition of eigenvalues,

$Ax_i = \lambda_i x_i$ for all $i = 1,2 \ldots n$.

Now, if we construct a matrix $X$ whose columns are eigenvectors $x_i$'s, on the right hand side each column of X has to be multiplied by the corresponding $\lambda_i$. In section 2.2.3 (scaling matrix), we have seen that this is equivalent to post multiplying $X$ with a diagonal matrix whose elements are $\lambda_i$'s. This essentially means, $AX = X\Lambda$. Since, eigenvectors in the columns of $X$ are linearly independent, $X$ is invertible. This concludes the proof $A = X\Lambda X^{-1}$.

**Theorem 2.7:** If a $n \times n$ square matrix $A$ has $n$ distinct eigenvalues, then it is diagonalizable.

**Proof:** In order to prove this, it will be sufficient to show that the matrix $A$ has $n$ linearly independent eigenvectors. Then by theorem 2.6, the matrix will be diagonalizable.

Let us assume the distinct eigenvalues of $n \times n$ square matrix $A$ as $\{\lambda_1, \lambda_2 \cdots\cdots \lambda_n\}$ and the corresponding eigenvectors as $\{x_1, x_2 \cdots\cdots x_n\}$. Fix an index $k$ for the eigenvalues and the corresponding eigenvectors. For $k = 1$, the independence of the lone eigenvector is evident. Let us assume that the eigenvectors are independent up to $k$. We will examine for $k + 1$. Suppose the following is true:

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k + \alpha_{k+1} x_{k+1} = 0 \tag{2.151}$$

Multiplying both the sides by matrix $A$ and using the relation $Ax_i = \lambda_i x_i$, we get

$$\alpha_1 \lambda_1 x_1 + \alpha_2 \lambda_2 x_2 + \cdots + \alpha_k \lambda_k x_k + \alpha_{k+1} \lambda_{k+1} x_{k+1} = 0 \tag{2.152}$$

Multiplying equation (2.151) with $\lambda_{k+1}$ and deducting from equation (2.152), we obtain:

$$\alpha_1 (\lambda_1 - \lambda_{k+1}) x_1 + \alpha_2 (\lambda_2 - \lambda_{k+1}) x_2 + \cdots + \alpha_k (\lambda_k - \lambda_{k+1}) x_k = 0 \tag{2.153}$$

Since, eigenvectors ($x_i$'s) are independent up to $k$, the above means all the coefficients are zero. Since, eigenvalues are also distinct, $(\lambda_i - \lambda_{k+1}) \neq 0$ for $i = 1,\ldots,k$. This means $a_1 = a_2 \cdots = a_k = 0$. Putting this in equation (2.151) shows, $a_{k+1} = 0$. This concludes the proof.

Note that theorem 2.6 requires $n$ independent eigenvectors and the eigenvalues need not be distinct. The second theorem (2.7) is more binding that way. The above two theorems can be combined to yield a definition of *non-defective* matrix (Theorem 2.8).

**Theorem 2.8**: A $n \times n$ square matrix $A$ is non-defective if and only if there exists a non-singular $X$ such that $A = X\Lambda X^{-1}$.

**Proof:** By definition of non-defective matrix in section, a $n \times n$ square matrix always has $n$ independent eigenvectors (even if some eigenvalues may be repeated). Then by theorem 3.7, it is proved.

The above theorem is valid in both $\Re^{n\times n}$ and $C^{n\times n}$.

In an $n$-dimensional vector space spanned by a set of $n$ linearly independent basis vectors, it is always possible to construct a set of $n$ orthonormal vectors which span the same subspace. This is done by a technique known as *Gram-Schmidt orthogonalization*. We will soon see that it plays an important role in computation of eigenvalues. We will prove the procedure for orthogonalization stated in Theorem 2.9 by construction, which will eventually serve as the algorithme for generating an orthogonal basis for the column space of any matrix.

**Theorem 2.9:** For a set of linearly independent vectors $\{x_1, x_2 \cdots x_n\}$ there exists an orthonormal set of vectors $\{y_1, y_2 \cdots y_n\}$ such that $\{x_1, x_2 \cdots x_k\}$ and $\{y_1, y_2 \cdots y_k\}$ span the same subspace for every $k \in \{1,2 \cdots n\}$.

**Proof:** We will first construct an orthonormal set of basis using *Gram Schmidt* orthogonalization procedure and then show that they span the same subspace.

Initialize $y_1$ as $y_1 = \dfrac{x_1}{\|x_1\|}$ and construct $y_k$ as:

$$z_{k+1} = x_{k+1} - \sum_{i=1}^{k} \left(x_{k+1}^T y_i\right) y_i \quad \text{and} \quad y_{k+1} = \frac{z_{k+1}}{\|z_{k+1}\|} \tag{2.154}$$

Notice, that all $y_i$'s are unit vectors. We need to test whether they are orthogonal, i.e., $y_i^T y_j = \delta_{ij}$. The $\delta_{ij}$ is the *Kronecker* delta defined as:

$$\delta_{ij} = \begin{cases} 0 \text{ for } i \neq j \\ 1 \text{ for } i = j \end{cases} \tag{2.155}$$

For $k = 1$, there is nothing to prove. Let us assume that the vectors are orthogonal up to $k$, *i.e.*, $\{y_1, y_2 \cdots y_k\}$ are orthogonal. Let us test for $k$+1,

$$y_{k+1}^T y_j = \frac{1}{\|z_{k+1}\|}\left(x_{k+1}^T y_j - \sum_{i=1}^{k}\left(x_{k+1}^T y_i\right) y_i^T y_j\right) = \frac{1}{\|z_{k+1}\|}\left(x_{k+1}^T y_j - x_{k+1}^T y_j\right) = 0 \tag{2.156}$$

This concludes the required proof, $y_{k+1}^T y_j = 0$ for $j = 1,2, \ldots k$.

Only thing left to show is that the space ($X_k$) spanned by $\{x_1, x_2 \cdots x_k\}$ and space ($Y_k$) spanned by $\{y_1, y_2 \cdots y_k\}$ are same for every $k \in \{1,2 \cdots n\}$. By definition of $y_1$, it is true for $k = 1$. Assume that it is true for $k$, i.e., $X_k = Y_k$ and let us examine for k+1.

The equation (2.154) shows that $x_{k+1}$ is a linear combination of $\{y_1, y_2 \cdots y_{k+1}\}$. This means $x_{k+1} \in Y_{k+1}$ and thus, $X_{k+1} \subset Y_{k+1}$.

The above relation also means $\dim X_{k+1} \le \dim Y_{k+1}$. Dimension of the vector space $X_{k+1}$, dim $X_{k+1} = k+1$. Since, there are only $k+1$ linearly independent vectors in $Y_{k+1}$, $\dim Y_{k+1} \le k+1$. Thus dim $X_{k+1}$ = dim $Y_{k+1}$. This essentially means $X_k = Y_k$ in light of the following proposition which we shall not prove here. Interested readers may look at any book on vector spaces for the proof.

**Proposition:** Let $X$ be a subspace of a $n$-dimensional vector space $Y$, i.e., $X \subset Y$, then $\dim X \le n$. If $\dim X = \dim Y = n$ then $X = Y$.

The *Gram-Schmidt* procedure gives us a way to replace the independent set of columns of any matrix $A$ with an orthonormal set of vectors spanning the same *column space*. We will now show that any $n \times n$ square matrix with $n$ linearly independent columns can be expressed as multiplication of a orthogonal matrix ($Q$) with orthonormal vectors as its columns and an upper triangular matrix $R$. Once again, we will show this by construction in theorem 2.10 which will also serve as the algorithme for carrying out such decomposition. We will illustrate the application of theorems 2.9 and 2.10 through example 2.17.

**Theorem 2.10:** If a $n \times n$ matrix $A$ has linearly independent columns then it can be decomposed as $A = QR$ where, $Q$ is a $n \times n$ matrix with orthonormal columns and $R$ is an $n \times n$ upper triangular matrix.

**Proof:** Let us denote the columns of $A$ as a linearly independent set of vectors $\{x_1, x_2 \cdots x_n\}$. Using Theorem 2.9, an orthonormal set of vectors $\{y_1, y_2 \cdots y_n\}$ can be constructed to span the same subspace as that spanned by the columns of $A$. Then, the composition of matrices $A$ and $Q$ can be written as the collection of column vectors as follows:

$$A = [x_1, x_2 \cdots x_n] \text{ and } Q = [y_1, y_2 \cdots y_n] \tag{2.157}$$

In order to prove the theorem, we need to now express $x_i$'s as linear combination of the $y_i$'s and show that the coefficients ($r_{ij}$) lead to an upper triangular matrix $R$. The relation $A = QR$ leads to the following relation between the columns of $A$ and $Q$,

$$x_j = \sum_{i=1}^{n} r_{ij} y_j \tag{2.158}$$

Multiplying both sides of the above equation by $y_i^T$ and using the relation $y_i^T y_j = \delta_{ij}$, we obtain, $r_{ij} = y_i^T x_j$. Since, the vectors $\{y_1, y_2 \cdots y_n\}$ can be computed from the columns of matrix $A$, $\{x_1, x_2 \cdots x_n\}$ using the Gram-Schmidt procedure, $r_{ij}$'s can be easily computed using vector products. However, we still need to show that $r_{ij}$'s lead to an upper triangular matrix in order to save computational effort by avoiding calculation of zero elements.

In order to visualize the upper triangular matrix, let us rearrange the equation (2.154) and change a few indices as follows:

$$x_j = z_j + \sum_{k=1}^{j-1} \left( x_j^T y_k \right) y_k \tag{2.159}$$

Now, $r_{ij}$'s can be written as:

$$r_{ij} = y_i^T x_j = y_i^T z_j + \sum_{k=1}^{j-1} \left( x_j^T y_k \right) y_i^T y_k = y_i^T y_j \left\| z_j \right\| + \sum_{k=1}^{j-1} \left( x_j^T y_k \right) y_i^T y_k \tag{2.160}$$

We notice in the above equation,

- the first term is non-zero only for $i=j$
- second term is non-zero for all $i=k$. Since, $k$ = 1,2, … ($j$-1), the second term is non-zero for $i$ = 1,2, … ($j$-1)

Combining the above two statements, we see that $r_{ij}$ is non-zero for $i$ = 1,2, … $j$ or $i \le j$. This constitutes an upper triangular matrix. We illustrate the relations as follows:

$$\begin{aligned}
x_1 &= r_{11} y_1 \\
x_2 &= r_{12} y_1 + r_{22} y_2 \\
x_3 &= r_{13} y_1 + r_{23} y_2 + r_{33} y_3 \\
&\bullet \\
&\bullet \\
x_n &= r_{1n} y_1 + r_{2n} y_2 + r_{3n} y_3 \cdots + r_{nn} y_n
\end{aligned} \tag{2.161}$$

**Example 2.17**: Perform a *QR* decomposition of the matrix given below,

$$A = \begin{bmatrix} 3 & 4 & 1 \\ 3 & 5 & 1 \\ 2 & 2 & 1 \end{bmatrix}$$

**Solution:** Denoting the columns of A as vectors $x_1$, $x_2$ and $x_3$ respectively, we obtain:

$$x_1 = \begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix}, \; x_2 = \begin{bmatrix} 4 \\ 5 \\ 2 \end{bmatrix} \text{ and } \; x_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

We will use Gram-Schmidt orthogonalization (2.154) to generate a set of orthonormal vectors.

$$\left\| x_1 \right\| = \sqrt{3^2 + 3^2 + 2^2} = 4.6904$$

$$y_1 = \frac{x_1}{\left\| x_1 \right\|} = \begin{bmatrix} 0.6396 \\ 0.6396 \\ 0.4264 \end{bmatrix}$$

$$z_2 = x_2 - \left(x_2^T y_1\right) y_1 = \begin{bmatrix} -0.2273 \\ 0.7727 \\ -0.8182 \end{bmatrix}$$

$$\|z_2\| = 1.1481, \quad y_2 = \frac{z_2}{\|z_2\|} = \begin{bmatrix} -0.1980 \\ 0.6730 \\ -0.7126 \end{bmatrix}$$

$$z_3 = x_3 - \left(x_3^T y_1\right) y_1 - \left(x_3^T y_2\right) y_2 = \begin{bmatrix} -0.1379 \\ 0.0690 \\ 0.1034 \end{bmatrix}$$

$$\|z_3\| = 0.1857, \quad y_3 = \frac{z_3}{\|z_3\|} = \begin{bmatrix} -0.7428 \\ 0.3714 \\ 0.5571 \end{bmatrix}$$

Now, we can constitute the $Q$ matrix with $y_1$, $y_2$ and $y_3$ as the columns,

$$Q = \begin{bmatrix} 0.6396 & -0.1980 & -0.7428 \\ 0.6396 & 0.6730 & 0.3714 \\ 0.4264 & -0.7126 & 0.5571 \end{bmatrix}$$

Elements of matrix $R$ can be computed using the relation $r_{ij} = y_i^T x_j$ as follows:

$r_{11} = y_1^T x_1 = 4.6904$, $r_{12} = y_1^T x_2 = 6.6092$, $r_{13} = y_1^T x_3 = 1.7056$, $r_{22} = y_2^T x_2 = 1.1481$, $r_{23} = y_2^T x_3 = -0.2375$ and $r_{33} = y_3^T x_3 = 0.1857$

The readers can check that the entries below the diagonal are indeed zero, i.e.,

$r_{21} = y_2^T x_1 = 0$, $r_{31} = y_3^T x_1 = 0$ and $r_{32} = y_3^T x_2 = 0$

Therefore, the matrix $R$ is,

$$R = \begin{bmatrix} 4.6904 & 6.6092 & 1.7056 \\ 0 & 1.1481 & -0.2375 \\ 0 & 0 & 0.1857 \end{bmatrix}$$

By definition of orthogonal matrices, $Q^T = Q^{-1}$. Thus, $A = QBQ^T$ means that matrices $A$ and $B$ are similar and share the same eigenvalues. Orthogonal matrices are important in the sense that every real matrix can be transformed to an upper triangular matrix by similarity transformation with an orthogonal matrix. *Schur* decomposition stated in theorem 2.11 ascertains this. A detail proof of the theorem is beyond the scope of this book. We will give an outline of the proof.

**Theorem 2.11:** For every real matrix $A$ with real eigenvalues, there is an orthogonal matrix $Q$ such that $A = QBQ^T$ where $B$ is an upper triangular matrix. If $A$ is complex, correspondingly there exist a unitary Q such that $A = QBQ^H$.

**Proof Outline in $\Re^n$ :** Denote an $n \times n$ matrix as $A_n$, an $(n\text{-}1) \times (n\text{-}1)$ matrix as $A_{n-1}$ and so on.

Determine one eigenvalue $\lambda_1$ and corresponding eigenvector $x_1$ for $A = A_n$. Chose other $(n\text{-}1)$ vectors $\{y_2 \cdots y_n\}$ such that $\{x_1, y_2 \cdots y_n\}$ forms an orthonormal basis for the $\Re^n$. Now, construct a matrix $X_n$. with $\{x_1, y_2 \cdots y_n\}$ as columns. Then, one can write,

$$X_n^T A X_n = \begin{bmatrix} \lambda_1 & a \\ 0 & A_{n-1} \end{bmatrix} \tag{2.162}$$

where, $a$ is some real number. One can repeat that with $A_{n-1}$ and obtain:

$$X_{n-1}^T A_{n-1} X_{n-1} = \begin{bmatrix} \lambda_2 & b \\ 0 & A_{n-2} \end{bmatrix} \tag{2.163}$$

where, $b$ is some real number. Now, if we define $Q_{n-1} = X_n \hat{X}_{n-1} = X_n \begin{bmatrix} 1 & 0 \\ 0 & X_{n-1} \end{bmatrix}$, we can write,

$$Q_{n-1}^T A Q_{n-1} = \begin{bmatrix} \lambda_1 & a_1 & a_2 \\ 0 & \lambda_2 & a_3 \\ 0 & 0 & A_{n-2} \end{bmatrix} \tag{2.164}$$

Proceeding similarly, we can write $Q^T A Q = B$ where, B is an upper triangular matrix with eigenvalues on the diagonal and $Q = X_n \hat{X}_{n-1} \hat{X}_{n-2} \cdots \hat{X}_2 \hat{X}_1$, a matrix with orthonormal vectors as columns.

In simple terms, the above theorem says that it is possible to reduce a real matrix $A$ to an upper triangular matrix $B$ by similarity transformation $B = Q^T A Q$ with orthogonal matrix $Q$ and thereby determining the eigenvalues of $A$ as the diagonal elements of $B$. Moreover, if the columns of matrix Q are eigenvectors of matrix A, it will be transformed to a diagonal matrix (Theorem 2.6). In summary, every real matrix through similarity transformation with an orthogonal matrix can be reduced to either a diagonal or an upper triangular matrix with eigenvalues on the diagonal. This forms the basis for the numerical method described in this section.

Finding the particular orthogonal matrix that reduces an arbitrary matrix $A$ to an upper triangular matrix $B$ or diagonal matrix $\Lambda$ is not easy. As with other numerical methods, we rely on generating a sequence of matrices through similarity transformation that eventually converges to $B$ or $\Lambda$.

Starting from the original square matrix, a sequence of similar matrices is generated by the application of similarity transformation with an orthogonal matrix. The orthogonal matrix ($Q$) obtained by decomposing the matrix $A$ into $QR$ at each step is used for the transformation.

The sequence converges to either an upper triangular matrix or a diagonal matrix with the eigenvalues on the diagonal. The preceding discussion makes it clear that whether a matrix becomes a diagonal or an upper triangular depends on the properties of the original matrix. The algorithm is summarized below:

$$\begin{aligned} A_0 &= A \\ A_k &= Q_k R_k \\ A_{k+1} &= Q_k^T A_k Q_k = Q_k^T Q_k R_k Q_k = R_k Q_k \end{aligned} \tag{2.165}$$

The iteration can be stopped when the change in the diagonal elements is below certain prescribed value. The change can be measure in absolute terms or as a relative percentage. A metric can be $\max_j \left| \frac{\lambda_j^{k+1} - \lambda_j^k}{\lambda_j^k} \right| \times 100$. We show an example for calculation of eigenvalues using the similarity transformation.

**Example 2.18:** Find the eigenvalues of the following matrix with a relative precision of 0.1% or less.

$$\begin{bmatrix} 3 & 4 & 1 \\ 3 & 5 & 1 \\ 2 & 2 & 1 \end{bmatrix}$$

**Solution:** We start with the original matrix and perform a *QR* decomposition. Compute the matrix *Q* using equation (2.154) and *R* using (2.159). The new matrix is calculated by multiplying *RQ*. This process is repeated until convergence. We tabulate the computation below:

| $k$ | $A_k = R_k Q_k$ | | | $Q_k$ | | | $R_k$ | | | $\varepsilon$ (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.0000 | 4.0000 | 1.0000 | 0.6396 | -0.1980 | -0.7428 | 4.6904 | 6.6092 | 1.7056 | |
| | 3.0000 | 5.0000 | 1.0000 | 0.6396 | 0.6730 | 0.3714 | 0.0000 | 1.1481 | -0.2375 | |
| | 2.0000 | 2.0000 | 1.0000 | 0.4264 | -0.7126 | 0.5571 | 0.0000 | 0.0000 | 0.1857 | |
| 2 | 7.9545 | 2.3043 | -0.0792 | 0.9968 | -0.0757 | -0.0257 | 7.9801 | 2.3703 | -0.0546 | |
| | 0.6331 | 0.9420 | 0.2941 | 0.0793 | 0.9765 | 0.2004 | 0.0000 | 0.7721 | 0.2723 | 866 |
| | 0.0792 | -0.1323 | 0.1034 | 0.0099 | -0.2018 | 0.9794 | 0.0000 | 0.0000 | 0.1623 | |
| 3 | 8.1420 | 1.7215 | 0.2166 | 1.0000 | -0.0078 | -0.0006 | 8.1423 | 1.7269 | 0.2199 | |
| | 0.0640 | 0.6990 | 0.4214 | 0.0079 | 0.9988 | 0.0482 | 0.0000 | 0.6863 | 0.4116 | 34.92 |
| | 0.0016 | -0.0328 | 0.1589 | 0.0002 | -0.0482 | 0.9988 | 0.0000 | 0.0000 | 0.1790 | |
| 4 | 8.1556 | 1.6505 | 0.2983 | 1.0000 | -0.0007 | 0.0000 | 8.1557 | 1.6509 | 0.2986 | |
| | 0.0055 | 0.6656 | 0.4442 | 0.0007 | 0.9999 | 0.0130 | 0.0000 | 0.6645 | 0.4416 | 11.08 |
| | 0.0000 | -0.0086 | 0.1788 | 0.0000 | -0.0130 | 0.9999 | 0.0000 | 0.0000 | 0.1845 | |
| 5 | 8.1568 | 1.6414 | 0.3199 | 1.0000 | -0.0001 | 0.0000 | 8.1568 | 1.6415 | 0.3199 | |
| | 0.0004 | 0.6587 | 0.4502 | 0.0001 | 1.0000 | 0.0036 | 0.0000 | 0.6587 | 0.4495 | 3.1 |
| | 0.0000 | -0.0024 | 0.1845 | 0.0000 | -0.0036 | 1.0000 | 0.0000 | 0.0000 | 0.1861 | |
| 6 | 8.1568 | 1.6398 | 0.3259 | 1.0000 | 0.0000 | 0.0000 | 8.1568 | 1.6398 | 0.3259 | |
| | 0.0000 | 0.6570 | 0.4519 | 0.0000 | 1.0000 | 0.0010 | 0.0000 | 0.6570 | 0.4517 | 0.88 |
| | 0.0000 | -0.0007 | 0.1861 | 0.0000 | -0.0010 | 1.0000 | 0.0000 | 0.0000 | 0.1866 | |
| 7 | 8.1569 | 1.6395 | 0.3276 | 1.0000 | 0.0000 | 0.0000 | 8.1569 | 1.6395 | 0.3276 | |
| | 0.0000 | 0.6565 | 0.4524 | 0.0000 | 1.0000 | 0.0003 | 0.0000 | 0.6565 | 0.4523 | 0.25 |
| | 0.0000 | -0.0002 | 0.1866 | 0.0000 | -0.0003 | 1.0000 | 0.0000 | 0.0000 | 0.1867 | |
| 8 | 8.1569 | 1.6394 | 0.3281 | | | | | | | |
| | 0.0000 | 0.6564 | 0.4525 | | | | | | | 0.07 |
| | 0.0000 | -0.0001 | 0.1867 | | | | | | | |

So, the eigenvalues are 8.1569, 0.6564 and 0.1867.

The reader should note that the eigenvectors cannot be determined in this way. By definition of similar matrices (Theorem 2.5), the eigenvectors change during similarity transform. However, once the eigenvalues are known, eigenvectors can be determined using any of the method for the solution of equation described in section 2.2.

The *QR* iteration scheme described in this section is really in the elementary form. Although, this is the basic procedure that can be employed to estimate the eigenvalues for small and medium size matrices, it is computationally inefficient for large matrices. The method described above can always be applied if computational efficiency is not a factor. With mildly defective matrices, this method may lead to accumulation of round-off errors. Some matrix preprocessing is often performed before applying *QR* iterations to economize computation for large matrices. Mainly, the matrix is reduced to upper Hessenberg form (Figure 2.7) prior to *QR* iteration. Detailed description of this reduction of a matrix to upper Hessenberg form is beyond the scope of this book. Interested readers may refer to Young and Gregory (1984) and Van Loan (1996) for detail. Van Loan (1996) is also a very good source for students interested in advanced methods for all aspects of matrix computations.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1k} & a_{1k+1} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots & a_{2k} & a_{2k+1} & \cdots & a_{2n} \\ 0 & a_{32} & a_{33} & a_{34} & \cdots & a_{3k} & a_{3k+1} & \cdots & a_{3n} \\ 0 & 0 & a_{43} & a_{44} & \cdots & a_{4k} & a_{4k+1} & \cdots & a_{4n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & a_{kk} & a_{kk+1} & \cdots & a_{kn} \\ 0 & 0 & 0 & 0 & \cdots & a_{k+1k} & a_{k+1k+1} & \cdots & a_{k+1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_{nn-1} & a_{nn} \end{bmatrix}$$

**Figure 2.7: A matrix in Hessenberg form or a quasi uppertriangular form. In this form, only one subdiagonal elements below the main diagonal may contain non-zero elements.**

# Exercise 2.3

1. Consider the following Matrix:

$$\begin{bmatrix} 7 & -2 & 1 \\ -2 & 10 & -2 \\ 1 & -2 & 7 \end{bmatrix}$$

a) Obtain the Equation of the Characteristic polynomial using Fadeev-Leverrier Method.
b) Solve the polynomial equation by Bairstow's method with a relative precision of 0.01% or less and thus, obtain all the eigenvalues.
c) Obtain all the eigenvalues by similarity transform.

d) Obtain the larges and smalles eigenvalues by Power method and inverse power method, respectively.
e) Compare the eigenvalues obtained using these methods.

2. Consider the following matrix:

$$\begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

a) Obtain all the eigenvalues using similarity transform.

b) Formulate the characteristic equation for the above matrix using Fadeev-Leverrier method.

3. Prove that the inverse power method with shift will converge to the eigenvector corresponding to the inverse of the eigenvalue nearest in magnitude to the shift applied.

4. For a square matrix $A$, show the following: (a) sum of the eigenvalues is equal to trace ($A$), (b) Product of the eigenvalues is equal to det ($A$).

5. Formulate the iteration matrix $S$ of Jacobi method (eq 2.112) and Gauss Seidel method (eq 2.113) for the Batman problem (eq 2.2). Compute the largest eigenvalues or spectral radii for both the iteration matrices. Explain why Jacobi method diverge and Gauss Seidel method converge. (Recall that the necessary condition is given by eq 2.122)

## 2.4 Summary

In the beginning of the chapter, Batman introduced the linear systems of equations to the readers and their usefulness for his new car. We started with providing some background on matrix algebra. Our intention was to give a visual sense of matrices to the readers. For this, we have introduced the matrices as a representation of Linear Transformation. To facilitate the readers, we have summarized the basic concepts of the vector space in a box. Various concepts such as, existence and uniqueness of the solution, eigenvalues, eigenvectors and matrix norms were logically introduced.

Presentation of numerical methods started with direct methods. Gauss elimination was presented as the basic fundamental method. All other methods such as Gauss Jordon, LU decomposition and Thomas Algorithme were derived using Gauss elimination. Stability of the methods was analyzed and matrix conditioning through pivoting, scaling and equilibration was introduced. A smooth transition to iterative methods was made through iterative improvement of solution using direct methods.

Two iterative methods namely Jacobi and Gauss Seidel were derived using a common framework which was also used for their convergence analysis. Gauss Seidel method was modified through rate of convergence analysis to yield SOR.

We presented eigenvalue estimation methods sequentially from single eigenvalue to multiple. Power method was derived in detail to estimate the largest eigenvalue. This framework was then used for the next two methods to estimate the smallest eigenvalue and the eigenvalue closest to any given value. At the end, we presented two methods for determining all the eigenvalues, one based on the solution of the characteristics polynomial and the other based

on similarity transformation. For the similarity transformation, the visualization of the matrix as linear transformation was useful once again.

Throughout the chapter, we attempted to present the methods in a logical sequence. All the methods presented were derived from the basic principles. A visualization was provided wherever possible. To this end, each method and key concepts were followed by examples. Wherever we left out vital detail, the readers were provided with appropriate references. References were provided for the advanced readers as well. At the end of each section, an exercise is given for the students to practice the lessons learned by solving some problems. Some excel templates and programs are provided in the included CD as well. Our goal will be fulfilled if this chapter helps to clarify basic concepts to an undergraduate student as well as serve as a launching pad for more advanced students. After linear equations, a natural progression will be to solve the non-linear equations. The following chapter deals with non-linear single equations and systems of non-linear equations. Watch out! Batman flies in to introduce it!