2. Numerical Linear Algebra

In this chapter we will study computational algorithms for solving problems in linear algebra. The study of these algorithms is important because 75% of all scientific problems require the solution of a system of linear equations. The problems that will be considered here include:

1. Solving a system of \( n \) linear equations in \( n \) variables (or, unknowns)
2. Solving an "over-determined" system of linear equations.

A system is said to be over-determined if the number of equations is greater than the number of variables.

2.1 System of Linear Equations

The problem we will study in this section is to solve the following system of equations for \( x_i, i = 1, 2, ..., n \).

\[
\begin{align*}
a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\
a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\
& \quad \vdots \\
a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n
\end{align*}
\]

(2.1)

(2.1) can also be written in the following form

\[ AX = b \]  

(2.2)

where \( A \) is an \( n \times n \) matrix, called the coefficient matrix, and \( X \) and \( b \) are \( n \)-dimensional vectors.

\[
A = [a_{i,j}] = \begin{bmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
& \ddots & \ddots & \ddots \\
& & \ddots & \ddots & \ddots \\
a_{n,1} & a_{n,2} & \cdots & a_{n,n}
\end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \quad \text{(2.3)}
\]

We start with the basic Gaussian elimination algorithm.

2.1.1 Gaussian Elimination without Pivoting
If $A$ in (2.2) is upper triangular, i.e., all the entries below the main diagonal are zero, then one can easily solve the system for $X$ by back substitution, as shown in the following example. Therefore, for any given $AX = b$, if $A$ is not upper triangular, we

1. first transfer the equations, by eliminating the lower part (entries below the main diagonal) of the coefficient matrix $A$, to an equivalent system which is in upper triangular form, and
2. then solve the upper triangular system by back substitution.

The first step is called the forward elimination step and the second step is called the back substitution step.

**Example** If the following system is given

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \\
-1x_2 + 6x_3 &= 6.1 \\
155x_3 &= 155
\end{align*}
\]
then $x_3 = 155/155 = 1$

\[
\begin{align*}
x_2 &= (6.1 - 6x_3)/(-1) = (6.1 - 6(1))/(-1) = .1/(-1) = -1 \\
x_1 &= (7 + 7x_2 - 0x_3)/10 = (7 + 7(-1) - 0(1))/10 = 0/10 = 0
\end{align*}
\]

This is an example of back substitution.

For example, if the following system is given,

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{......... (1)} \\
-3x_1 + 2x_2 + 6x_3 &= 4 \quad \text{......... (2)} \\
5x_1 - x_2 + 5x_3 &= 6 \quad \text{......... (3)}
\end{align*}
\]

which is equivalent to

\[
\begin{bmatrix}
10 & -7 & 0 \\
-3 & 2 & 6 \\
5 & -1 & 5
\end{bmatrix}
\begin{bmatrix}
x_1 \\ x_2 \\ x_3
\end{bmatrix}
=
\begin{bmatrix}
7 \\ 4 \\ 6
\end{bmatrix}
\quad \text{(2.5)}
\]

one can perform the forward elimination step and the back substitution step as follows.

**Forward Elimination**

- To eliminate the entries below 10 in (2.5), first subtract $(-3/10)$ times equation (1) from equation (2) in (2.4), then subtract $5/10$ times equation (1) from equation (3) in (2.4). The first subtraction eliminates the coefficient of $x_1$ in equation (2) and the second subtraction
eliminates the coefficient of \( x_1 \) in equation (3). The equivalent system one gets after these subtractions is as follows. Note that equation (1) in (2.4) is unchanged.

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{(1)} \\
-0.1x_2 + 6x_3 &= 6.1 \quad \text{(2)} \\
2.5x_2 + 5x_3 &= 2.5 \quad \text{(3)}
\end{align*}
\]

To eliminate the coefficient of \( x_2 \) in equation (3) of (2.6), subtract \((2.5/-1)\) times equation (2) from equation (3) in (2.6). The equivalent system one gets after this subtraction is as follows. Note that equations (1) and (2) in (2.6) are unchanged.

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{(1)} \\
-0.1x_2 + 6x_3 &= 6.1 \quad \text{(2)} \\
155x_3 &= 155 \quad \text{(3)}
\end{align*}
\]

**Back Substitution**

- (2.7), an upper triangular system, can be solved by back substitution as follows.

\[
\begin{align*}
x_3 &= \frac{155}{155} = 1 \\
x_2 &= \frac{6.1 - 6x_3}{-1} = \frac{6.1}{-1} = -1 \\
x_1 &= \frac{7 - 0x_3 + 7x_2}{10} = \frac{7}{10} = 0
\end{align*}
\]

The numbers \((-3/10) = -0.3, (5/10) = 0.5, (2.5/-1) = -2.5\) used in the forward elimination step are called the *multipliers*. The divisor used to form each multiplier is called the *pivot*.

A system of \( n \) linear equations with \( n \) variables \( A \mathbf{X} = \mathbf{b} \), as the one shown in (2.1), is usually stored in an \( n \times (n+1) \), 2-dimensional array \( W \), with the coefficients \( a_{i,j} \) stored in entries \((i,j)\) of \( W \) for \( 1 \leq i, j \leq n \) and the constants \( b_i \) stored in entries \((i, n+1)\) of \( W \) for \( 1 \leq i \leq n \). This storage mode allows operations on \( \mathbf{b} \) to be carried out as extensions of operations on \( A \).

An algorithm for the above naive approach can be given as follows. The input to this algorithm, called Gaussian Elimination without Pivoting, is an \( n \times (n+1) \) array \( W \) where the coefficient matrix \( A \) and the vector \( \mathbf{b} \) of a given system of linear equations \( A \mathbf{X} = \mathbf{b} \) are stored in the above mode. The output of this algorithm is an 1-dimensional vector \( \mathbf{X} \) where the solution of the given system of linear equations is stored. Entry \((i,j)\) of \( W \) will be referred as \( W_{i,j} \) and entry \( i \) of \( \mathbf{X} \) will be referred as \( \mathbf{X}_i \).
Algorithm: Gaussian Elimination without Pivoting (GEWP)

1. [Forward Elimination]
   for $i := 1$ to $n-1$ do
   for $j := i+1$ to $n$ do
     $W_{j,i} \leftarrow m_{j,i} = W_{j,i} / W_{i,i}$;
   for $k := i+1$ to $n+1$ do
     $W_{j,k} \leftarrow W_{j,k} - m_{j,i} * W_{i,k}$;

2. [Back Substitution]
   $X_n \leftarrow W_{n,n+1} / W_{n,n}$;
   for $k := n-1$ downto 1 do
     $X_k \leftarrow (W_{k,n+1} - \sum_{j=k+1}^{n} W_{k,j} * X_j) / W_{k,k}$;

Note that (1) technically, $W_{i,k}, i = k+1, ..., n$, should have a zero stored. Instead of this, we store the multipliers $m_{i,k}$ for possible use later on. (2) $X_k, k = n, n-1, ..., 1$, can be stored in $W_{k,n+1}$ to save memory.

2.1.2 Gaussian Elimination with Partial Pivoting

Consider the following example.

\[
\begin{align*}
0x_1 + x_2 + x_3 &= 1 \quad \ldots \ldots \ (1) \\
x_1 + 2x_2 + 3x_3 &= 4 \quad \ldots \ldots \ (2) \\
2x_1 - x_2 - x_3 &= 1 \quad \ldots \ldots \ (3)
\end{align*}
\]

In this case, since the coefficient of $x_1$ in the first equation is zero, we can not compute the multipliers $m_{2,1}$ and $m_{3,1}$ in the Gaussian elimination process. A remedy is to interchange equation (1) with equation (2) such as follows and then start.

\[
\begin{align*}
x_1 + 2x_2 + 3x_3 &= 4 \quad \ldots \ldots \ (1) \\
0x_1 + x_2 + x_3 &= 1 \quad \ldots \ldots \ (2) \\
2x_1 - x_2 - x_3 &= 1 \quad \ldots \ldots \ (3)
\end{align*}
\]
Now, consider another example, assume 5 decimal digits in each number and chopped arithmetic.

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{(1)} \\
-3x_1 + 2.099x_2 + 6x_3 &= 3.901 \quad \text{(2)} \\
5x_1 - x_2 + 5x_3 &= 6 \quad \text{(3)}
\end{align*}
\]

In the process of Gaussian elimination for the first column, we would get \( m_{2,1} = -3/10 = -0.3 \) and \( m_{3,1} = 5/10 = 0.5 \), and the system becomes

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{(1)} \\
-0.001x_2 + 6x_3 &= 6.001 \quad \text{(2)} \quad \text{(2.8)} \\
2.5x_2 + 5x_3 &= 2.5 \quad \text{(3)}
\end{align*}
\]

There is no error generated in this step. Now, by continuing the Gaussian elimination process for the second column, with equation (2) as the pivot equation, we would get \( m_{3,2} = 2.5/(-0.001) = -2500 \), and the system becomes

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{(1)} \\
-0.001x_2 + 6x_3 &= 6.001 \quad \text{(2)} \quad \text{(2.9)} \\
15005x_3 &= 15004 \quad \text{(3)}
\end{align*}
\]

By performing back substitution, we then get

\[
x_3 = 15004/15005 = .99993
\]

\[
x_2 = (6.001 - 6x_3)/(-0.001) = (6.001 - 6*(.99993))/(-0.001)
\]

\[
= (6.0010 - 5.9985)/(-0.001) = -1.5
\]

\[
x_1 = (7 + 7x_2)/10 = (7 + 7*(-1.5))/10 = -.35
\]

However, the exact solution is \((x_1 = 0, x_2 = -1, x_3 = 1)\). The reason that we get big round-off errors in this case is because of \( m_{3,2} \). Note that the value of \( m_{3,2} \) is quite large. When one has such a large multiplier, it is quite possible for one to get big round-off error since large multiplier tends to exaggerate round-off error. Indeed, in the above case, the constant term on the right side of equation (3) in (2.9) should have been 15005. But because of the large value of \( m_{3,2} \), we get 15004 instead. A remedy is to interchange equations (2) and (3) in the Gaussian elimination process for the second column, so we would get smaller value for \( m_{3,2} \).
These examples show that, when performing the Gaussian elimination process, equations should be interchanged not just to avoid zero entries on the main diagonal, but also to avoid generating large multipliers. In general, it is desirable to have multipliers smaller than or equal to 1 in magnitude so propagation of round-off errors can be minimized. This is accomplished by doing partial pivoting defined as follows in the Gaussian elimination phase:

"If the coefficient of $x_k$ in equations $k+1$, $k+2$, ..., $n$ are to be eliminated, we first scan the $k$th column of the matrix from the diagonal down to find the element that is largest in magnitude. The row (equation) containing this element is then interchanged with the $k$th row (equation) and is used as the pivot equation."

For instance, in the above example, before performing the elimination process for the second column, one should interchange equations (2) and (3) in (2.8) to get a system as follows.

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{......... (1)} \\
2.5x_2 + 5x_3 &= 2.5 \quad \text{......... (2)} \\
-0.001x_2 + 6x_3 &= 6.001 \quad \text{......... (3)}
\end{align*}
\]

Then by using the new equation (2) as the pivot equation, we get the following equivalent system.

\[
\begin{align*}
10x_1 - 7x_2 + 0x_3 &= 7 \quad \text{......... (1)} \\
2.5x_2 + 5x_3 &= 2.5 \quad \text{......... (2)} \\
6.002x_3 &= 6.002 \quad \text{......... (3)}
\end{align*}
\]

Now, by performing back substitution, we get the exact solution, as follows.

\[
\begin{align*}
x_3 &= 1 \\
x_2 &= (2.5 - 5* x_3)/2.5 = (2.5 - 5)/2.5 = -1 \\
x_1 &= (7 + 7* x_2)/10 = (7 - 7)/10 = 0
\end{align*}
\]

However, one should not physically interchange equations in the elimination process since it takes too much time. One should use a pivot vector $P$ to record the pivot strategy, instead. $P$ is a vector of $n$ components whose $i$th component serves as a pointer to the physical location of the pivot equation for column $i$ ($i$th elimination stage). Initially, $P[i]$ is set to $i$ for $i = 1, 2, ..., n$ where $P[i]$ is the $i$th component of $P$. If, in the $k$th elimination stage ($1 \leq k \leq n-1$), row $j$ ($k < j \leq n$) is required to be interchanged with row $k$ and used as the pivot equation for this stage, one simply interchange the contents of $P[j]$ and $P[k]$. Therefore, $P[k]$ points to the physical location of the pivot equation for this stage. If a multiplier $m_{k,l}$ is to be computed for
some $j > k$, one should use coefficients of $x_k$ stored in row $P[k]$ and row $P[l]$.

An algorithm solving a system of equations using Gaussian elimination with partial pivoting is given as follows. The input to this algorithm, called Gaussian Elimination with Partial Pivoting, is an $n \times (n+1)$ array $W$ and an $n$-entry array $P$. The coefficient matrix $A$ and the vector $b$ of the given system of linear equations $AX = b$ are stored in $W$ in the mode described in previous section. $P$ will be serving as the pivot vector. The output of this algorithm is an 1-dimensional vector $X$ where the solution of the given system of linear equations is stored. Entry $(i,j)$ of $W$ will be referred as $W[i,j]$ and entry $i$ of $X$ and $P$ will be referred as $X[i]$ and $P[i]$.

---

**Algorithm: Gaussian Elimination with Partial Pivoting (GEWPP)**

1. **[Initialization]**
   
   for $i := 1$ to $n$ do
   
   $P[i] \leftarrow i$;

2. **[Forward Elimination]**
   
   for $i := 1$ to $n-1$ do
   
   find $k \geq i$ such that $W[P[k], i] = \max \{ W[P[j],i] \mid i \leq j \leq n \}$;
   
   if ($k <> i$) then
   
   $P[k] \leftarrow P[i]$;

   for $j := i+1$ to $n$ do
   
   $W[P[j], i] \leftarrow m = W[P[j],i] / W[P[i],i]$;

   for $k := i+1$ to $n+1$ do
   
   $W[P[j], k] \leftarrow W[P[j], k] - m * W[P[i], k]$;

3. **[Back Substitution]**

   $X[n] \leftarrow W[P[n], n+1] / W[P[n], n]$;

   for $k := n-1$ downto 1 do
   
   $X[k] \leftarrow (W[P[k], n+1] - \sum_{j=k+1}^{n} W[P[k], j] * X[j]) / W[P[k], k]$;

---

2.1.3 Errors

In this section we will look at the numerical validity of the Gaussian elimination algorithm, i.e., ascertain the error of a computed (or, approximate) solution.

Given a system of linear equations $AX = b$. Let $\hat{X}$ denote the computed solution (using finite precision arithmetic). To measure the error of $\hat{X}$, one needs to compute

$$e \equiv X - \hat{X} \quad (2.10)$$
Unfortunately, this item usually can not be computed directly since the exact solution $X$ is not known to us (otherwise, we wouldn’t have to solve $AX = b$). An alternative is to see how well $\hat{X}$ satisfies the original equations. This can be done by computing $A\hat{X}$ and then measuring the difference, $r$, between $b$ and $A\hat{X}$

$$r \equiv b - A\hat{X}$$

(2.11)

$r$ is called the residual vector of $\hat{X}$. Since $r$ and $e$ satisfy the following equation,

$$r = b - A\hat{X} = A(X - \hat{X}) = Ae$$

it seems that one can use $r$ as an indicator on the size of $e$. Indeed, $r$ equals zero if the computed solution $\hat{X}$ is the exact solution. Unfortunately, a small $r$ does not necessarily imply a small $e$, as evidenced by the following example.

$$\begin{align*}
3.02x_1 - 1.05x_2 + 2.53x_3 &= -1.61 \\
4.33x_1 + .56x_2 - 1.78x_3 &= 7.23 \\
-.83x_1 - .54x_2 + 1.47x_3 &= -3.38
\end{align*}$$

If the above system of linear equations is solved using GEWPP with three decimal digits in each floating-point number and chopped arithmetic, we get the following computed solution

$$\hat{X} = \begin{bmatrix} .880 \\ -2.35 \\ -2.66 \end{bmatrix}$$

Therefore, we have

$$A\hat{X} = \begin{bmatrix} -1.6047 \\ 7.2292 \\ -3.3716 \end{bmatrix} \quad \text{and} \quad r = b - A\hat{X} = \begin{bmatrix} -.0053 \\ .0008 \\ -.0084 \end{bmatrix}$$

The residual vector is quite small. But $\hat{X}$ carries a relatively large error since the exact solution of this system is $X = (1, 2, -1)^T$.

In general, numerical methods can only be expected to produce a solution $\hat{X}$ with "small" residual vector. Hence, guidelines should be developed so that one can tell when the computed solution will not be good and, in such a case, what one should do to improve the accuracy of the computed solution.

To address these problems, we need to compare the "sizes" of vectors and matrices. The size of a vector $X = (x_1, x_2, ..., x_n)^T$ is usually measured in terms of "norm" defined as follows.
\[ \|X\| = (x_1^2 + x_2^2 + \cdots + x_n^2)^{1/2} = (X^T X)^{1/2} \] (2.12)

\|X\| is also called the Euclidean length of \(X\). The vector norm satisfies the following properties:

1. \(\|X\| \geq 0\) for any \(X\), and \(\|X\| = 0\) if and only if \(X = 0\).
2. \(\|\alpha X\| = |\alpha| \|X\|\) for any scalar \(\alpha\) and any vector \(X\).
3. \(\|X + Y\| \leq \|X\| + \|Y\|\) for all vectors \(X\) and \(Y\).

The third property is also called the triangular inequality property. The proof of these properties can be found in most books in linear algebra.

Corresponding to an \(n\)-dimensional vector norm, we can define a norm for \(n \times n\) matrices as follows. Given an \(n \times n\) matrix \(A\), the norm of the matrix is defined as

\[ ||A|| \equiv \max_{\|X\|=1} \frac{||AX||}{\|X\|} = \max \frac{||AX||}{\|X\|} \] (2.13)

where \(X\) and \(AX\) are \(n\)-dimensional vectors and \(\|X\|\) and \(\|AX\|\) are vector norm for \(n\)-dimensional vectors. With this definition of matrix norm, it is easy to see that for any \(n\)-dimensional vector \(X\), we have

\[ ||AX|| \leq ||A||^* ||X|| \] (2.14)

and for some \(X\), we have

\[ ||AX|| \geq ||A||^* ||X|| \]

In general, matrix norms satisfy the following properties:

1. \(||A|| \geq 0\) for any \(A\), and \(||A|| = 0\) if and only if \(A = 0\).
2. \(||\alpha A|| = |\alpha|^* ||A||\) for any scalar \(\alpha\) and any matrix \(A\).
3. \(||A + B|| \leq ||A|| + ||B||\) for any two matrices \(A\) and \(B\).
4. \(||AB|| \leq ||A||^* ||B||\) for any two matrices \(A\) and \(B\).

The proof of these properties can also be found in most books in linear algebra.

We are ready to address the relationship between and error \(e = X - \hat{X}\) and the residual vector \(r = b - A \hat{X}\) now. We need the following lemma in the proof of our main result.

**Lemma 1** If \(e\) and \(r\) are defined as in (2.10) and (2.11) for a given system of linear equations \(AX = b\) whose coefficient matrix \(A\) has an inverse matrix \(A^{-1}\) then we have

\[ \frac{||r||}{||A||} \leq ||e|| \leq ||A^{-1}||^* ||r|| \] (2.15)
where $||e||$ and $||r||$ are vector norms, $||A||$ and $||A^{-1}||$ are matrix norms.

Proof Since $r = Ae$, it follows from (2.14) that $||r|| \leq ||A||^* ||e||$
and, consequently,

$$\frac{||r||}{||A||} \leq ||e||.$$

On the other hand, since $A^{-1}r = e$, it follows from (2.14) again that $||e|| \leq ||A^{-1}||^* ||r||$

Hence, by combining the above two inequalities, we get (2.15).

Lemma 1 shows that the error $e$ carried by the computed solution $\hat{X}$ of a system of linear equations $AX = b$ is bounded above by the norm of $A^{-1}$ times the norm of the residual vector $r$. Hence, when $||A^{-1}||$ is large, even the residual vector is small, one could still have a large $e$. However, we are more interested in the relative error than the absolute error. The following theorem shows the relationship between the size of the error relative to the solution and the size of the residual vector relative to $b$.

**Theorem 3** If $e$ and $r$ are defined as in (2.10) and (2.11) for a given system of linear equations $AX = b$ whose coefficient matrix $A$ has an inverse matrix $A^{-1}$ then we have

$$\frac{1}{||A||^* ||A^{-1}||} \frac{||r||}{||b||} \leq \frac{||e||}{||X||} \leq \left(\frac{||A||^* ||A^{-1}||}{||A||^* ||A^{-1}||}ight) \frac{||r||}{||b||}$$

(2.16)

where $||e||$, $||X||$, $||r||$, $||b||$ are vector norms, and $||A||$ and $||A^{-1}||$ are matrix norms.

Proof Since $AX = b$, following the technique used in the proof of Lemma 1, we can prove that

$$\frac{||b||}{||A||} \leq ||X|| \leq ||A^{-1}||^* ||b||$$

Consequently,

$$\frac{1}{||A^{-1}||^* ||b||} \leq \frac{1}{||X||} \leq \frac{||A||}{||X|| ||b||}$$

By multiplying (2.5) by the above inequality, we then have (2.16).
$\|A\| \leq \|A^{-1}\|$ is called the condition number of $A$ and is denoted by $\text{cond}(A)$. Hence (2.16) can also be written as

$$\frac{1}{\text{cond}(A)} \frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|X\|} \leq \text{cond}(A) \frac{\|r\|}{\|b\|} \tag{2.17}$$

This inequality provides us with a means to estimate the relative error. It is possible to make $\|r\|/\|b\|$ very small by increasing the accuracy of the machine. However, if the condition number of $A$ is very large then the computed solution may still have large relative error. Hence, when the condition number of the coefficient matrix $A$ is large, the residual vector gives little information about the accuracy of $\hat{X}$. The relative residual is a good measure of the relative error of $\hat{X}$ only if the condition number is close to 1. A coefficient matrix (system) is said to be ill-conditioned if the matrix has large condition number. In the following we will show an example of an ill-conditioned system.

Consider two intersecting lines $l_1$ and $l_2$ defined in the following figure whose slopes are almost identical. Without loss of generality, we shall assume that these lines are represented in normal form, i.e.,

$$(a_{11})^2 + (a_{12})^2 = (a_{21})^2 + (a_{22})^2 = 1 \tag{2.18}$$

Since the slope of $l_1$ is close to the slope of $l_2$, we have

$$\frac{a_{11}}{a_{12}} \approx \frac{a_{21}}{a_{22}}$$

It is easy to see that, in this case, row one and row two of the following system’s coefficient matrix will be almost identical.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$$

If one attempts to solve this system of equations for $X=(x_1, x_2)^T$, one is bound to get large round-off error since one will get severe cancellation when updating the coefficient of $x_2$ in
equation 2. Nevertheless, if the computed solution is $\hat{X}$ then the residual vector is

$$r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = b - A \hat{X} = \begin{bmatrix} b_1 - (a_{11}\hat{x}_1 + a_{12}\hat{x}_2) \\ b_2 - (a_{21}\hat{x}_1 + a_{22}\hat{x}_2) \end{bmatrix}$$

Note that $r_1$ is the distance from $\hat{X}$ to the line $l_1$, $r_2$ is the distance from $\hat{X}$ to the line $l_2$. Hence $r$ could be very small even though $\hat{X}$ is quite far away from the exact solution $X$. In this case, the system is ill-conditioned. The system will be well-conditioned if the lines are almost perpendicular to each other.

The question that remains to be answered is: if we have an ill-conditioned system, what can we do and what should we do so that the accuracy of the computed solution can be improved? We will answer this question in the next section.

### 2.1.4 Iterative Improvement

In this section we will give an iterative procedure to improve the accuracy of the computed solution $\hat{X}$ of a given system of linear equations $AX = b$.

Note that the computed solution $\hat{X}$ has an error $e = X - \hat{X}$ which satisfies the following equation.

$$r = b - A \hat{X} = A e$$

Therefore, if we can solve $A e = r$ to get $e$ then the exact solution $X$ of $AX = b$ is simply the sum of $e$ and $\hat{X}$. The computed solution $\hat{e}$ of $A e = r$ usually, however, would carry error too. Hence, the sum of $\hat{X}$ and $\hat{e}$ will not be the exact solution $X$. But it will be closer to $X$ than $\hat{X}$ Nevertheless, we can repeat the same process to $\hat{X} + \hat{e}$ to improve the accuracy of the solution. This leads to the following iterative procedure to improve the accuracy of the computed solution of $AX = b$.

1. Solve $AX = b$ (single precision) to get $\hat{X}^{(1)}$
2. Compute $r^{(1)} = b - A \hat{X}^{(1)}$ (double precision)
3. Solve $A e = r^{(1)}$ (single precision) to get $\hat{e}^{(1)}$ and set $\hat{X}^{(2)} = \hat{X}^{(1)} + \hat{e}^{(1)}$
4. Compute $r^{(2)} = b - A \hat{X}^{(2)}$ (double precision)
5. Solve $A e = r^{(2)}$ (single precision) to get $\hat{e}^{(2)}$ and set $\hat{X}^{(3)} = \hat{X}^{(2)} + \hat{e}^{(2)}$
6. Compute $r^{(3)} = b - A \hat{X}^{(3)}$ (double precision)
7. Solve $A e = r^{(3)}$ (single precision) to get $\hat{e}^{(3)}$ and set $\hat{X}^{(4)} = \hat{X}^{(3)} + \hat{e}^{(3)}$

.....

The iteration continues until $\|\hat{e}^{(k)}\| \approx 10^{-t}$ if $t$ decimal digits are carried during the calculation.
The reason that double precision is required in the even steps is because the process of computing \( r \) involves severe cancellation. Extra precision must be used in computing \( r \). Otherwise, the iterative procedure may fail to generate more accurate result.

Steps (3), (5), (7), ... can be solved using the LU decomposition of \( PA \) where \( P \) is the permutation vector. That is, using forward elimination to solve

\[
L \mathbf{d} = P \mathbf{r}^{(i)}
\]

for \( \mathbf{d} \) first, then using back-substitution to solve the following equation for \( \hat{\mathbf{e}}^{(i)} \).

\[
U \mathbf{e} = \mathbf{d}
\]