

Module 4 – Linear Systems of Equations – Part II

Lesson goals

1. Understanding that LU factorization involves decomposing the coefficient matrix into two triangular matrices that can then be used to efficiently evaluate different right-hand-side vectors.
2. Knowing how to express Gauss elimination as an LU factorization.
3. Given an LU factorization, knowing how to evaluate multiple right-hand-side vectors.
4. Knowing how to determine the matrix inverse in an efficient manner based on LU factorization.
5. Understanding how the matrix inverse can be used to assess stimulus-response characteristics of engineering systems.
6. Understanding the meaning of matrix and vector norms and how they are computed.
7. Knowing how to use norms to compute the matrix condition number.
8. Understanding how the magnitude of the condition number can be used to estimate the precision of solutions of linear algebraic equations.

Introduction

Gauss elimination is designed to solve systems of linear algebraic equations:

$$Ax = b$$

Although it certainly represents a sound way to solve such systems, it becomes inefficient when solving equations with the same coefficients A , but with different right-hand-side vector b , specifically in the large setting.

LU factorization methods separate the time-consuming elimination of the matrix A from the manipulations of the right-hand side b . Thus, once A has been factored or decomposed, multiple right-hand-side vectors can be evaluated in an efficient manner.

Suppose that A has been factored into the triangular form $A = LU$, where L is lower triangular and U is upper triangular. Then we can solve for x more easily by using a two-step process.

- We let $y = Ux$ and solve the lower triangular system $Ly = b$ for y . Since L is triangular, determining y from this equation requires only $O(n^2)$ operations.
- Once y is known, the upper triangular system $Ux = y$ requires only an additional $O(n^2)$ operations to determine the solution x .

Solving a linear system $Ax = b$ in factored form means that the number of operations needed to solve the system $Ax = b$ is reduced from $O(n^3/3)$ to $O(2n^2)$.

Question: How to compute matrices L and U in LU decomposition?

LU decomposition from Gaussian elimination

Suppose that Gaussian elimination can be performed on the system $Ax = b$ without row interchanges. This is equivalent to having nonzero pivot elements $a_{ii}^{(i-1)}$, for $i = 1, 2, \dots, n$, assuming that $a_{11}^{(0)} = a_{11}$. By Gaussian elimination, the linear systems of equations

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}$$

becomes

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_3 \\
&\vdots \\
a^{(n-1)}_{nn}x_n &= b^{(n-1)}_n
\end{aligned}$$

Now, let us define

$$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2n} \\ 0 & 0 & a''_{33} & \cdots & a''_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a^{(n-1)}_{nn} \end{bmatrix}$$

We also define f_{ij} 's as below:

$$f_{ij} = \frac{a^{(j-1)}_{ij}}{a^{(j-1)}_{jj}}, \quad \text{for } i = 2, \dots, n; j = 1, \dots, i-1$$

For example:

$$f_{21} = \frac{a_{21}}{a_{11}}, \quad f_{31} = \frac{a_{31}}{a_{11}}, \dots, f_{n1} = \frac{a_{n1}}{a_{11}}$$

$$f_{32} = \frac{a'_{32}}{a'_{22}}, \quad f_{42} = \frac{a'_{42}}{a'_{22}}, \dots, f_{n2} = \frac{a'_{n2}}{a'_{22}}$$

and so on

Then, we define the matrix L as below:

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ f_{21} & 1 & 0 & \cdots & 0 \\ f_{31} & f_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & f_{n3} & \cdots & 1 \end{bmatrix}$$

By the Gaussian elimination procedure, it is shown that with the matrices L and U defined above, it yields

$$A = LU$$

which is called the LU factorization (decomposition). For simplicity, assume that

$$L = [l_{ij}]_{n \times n} \quad \text{and} \quad U = [u_{ij}]_{n \times n}$$

Once the matrix factorization is complete, the solution to a linear system of the form $Ax = LUx = b$ is found by first letting $y = Ux$ and solving $Ly = b$ for y . Since L is lower triangular, we have

$$Ly = b \Rightarrow \begin{cases} y_1 = \frac{b_1}{l_{11}} = b_1 \\ y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right), \quad i = 2, \dots, n \end{cases}$$

$\begin{cases} l_{11} y_1 = b_1 \\ l_{21} y_1 + l_{22} y_2 = b_2 \\ l_{31} y_1 + l_{32} y_2 + l_{33} y_3 = b_3 \end{cases}$

After y is found by this forward-substitution process, the upper-triangular system $Ux = y$ is solved for x by backward substitution using the equations

$$Ux = y \Rightarrow \begin{cases} x_n = \frac{y_n}{u_{nn}} \\ x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{j=i+1}^n u_{ij} x_j \right), \quad j = n-1, n-2, \dots, 1 \end{cases}$$

$$\begin{cases} u_{n-1, n-1} x_{n-1} + u_{n-1, n} x_n = y_{n-1} \longrightarrow \\ u_{nn} x_n = y_n \longrightarrow x_n = \frac{y_n}{u_{nn}} \end{cases}$$

Example. Use Gaussian elimination method to find the LU factorization of the coefficient matrix of the following system and then solve it by LU factorization method.

$$\begin{cases} 3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 = -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 = 71.4 \end{cases}$$

For this example, you must obtain matrices L and U are as follows:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0.033333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{bmatrix}$$

$$U = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix}$$

Step 1. Solve $Ly = b$ for y .

$$Ly = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333 & 1 & 0 \\ 0.1 & -0.0271 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{bmatrix} \Rightarrow \begin{cases} y_1 = 7.85 \\ y_2 = -19.3 - 0.0333 \times 7.85 \\ \quad \rightarrow y_2 = -19.5614 \\ y_3 = 71.4 - 0.1 \times 7.85 \\ \quad - (-0.0271)(-19.5614) \\ \rightarrow y_3 = 70.0849 \end{cases}$$

Step 2. Solve $Ux = y$ for x .

$$Ux = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.0033 & -0.2933 \\ 0 & 0 & 10.0120 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7.85 \\ -19.5614 \\ 70.0849 \end{bmatrix} \Rightarrow \begin{cases} 3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \\ 7.0033x_2 - 0.2933x_3 = -19.5614 \\ 10.012x_3 = 70.0849 \end{cases}$$

3rd eqn. $x_3 = 7.0001$

2nd eqn. $7.0033x_2 = -19.5614 + 0.2933(7.0001) \Rightarrow x_2 = -2.5$

1st eqn. $\Rightarrow 3x_1 = 7.85 + 0.1(-2.5) + 0.2(7.0001) \Rightarrow x_1 = 3.0$

LU decomposition with pivoting

Just as for Gauss elimination, partial pivoting is necessary to obtain reliable solutions with LU factorization. If we knew the row interchanges that were required to solve the system by Gaussian elimination, we could arrange the original equations in an order that would ensure that no row interchanges are needed.

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

Hence there is a rearrangement of the equations in the system that permits Gaussian elimination to proceed without row interchanges. This implies that for any nonsingular matrix A , a permutation matrix P exists for which the system

matrix A is called orthogonal

$$\text{if } A_i \cdot A_j = 0$$

for all $i \neq j$, where

A_j is the j^{th} column of the matrix.

can be solved without row interchanges. Therefore, this matrix PA can be factored into

$$PA = LU$$

where L is lower triangular and U is upper triangular. Because $P^{-1} = P^T$, this produces the factorization

$$A = P^{-1}LU = (P^T L)U$$

$$PA = LU$$

$$\rightarrow P^{-1}(PA) = P^{-1}LU$$

$$\rightarrow A = P^{-1}LU$$

$$P^{-1} = P^T \rightarrow \boxed{A = P^T LU}$$

The matrix U is still upper triangular, but $P^T L$ is not lower triangular unless $P = I$.

Example. Compute the LU factorization and find the solution for the following systems of linear equations (it requires partial pivoting)

$$\begin{aligned} 0.0003x_1 + 3.0000x_2 &= 2.0001 \\ 1.0000x_1 + 1.0000x_2 &= 1.0000 \end{aligned}$$

$$L = \begin{bmatrix} 0.0003 & 1 \\ 1 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 1 \\ 0 & 2.9997 \end{bmatrix}$$

$$A^{-1} = A^T$$

Step 1. Solve $Ly = b$ for y .

$$Ly = b \Rightarrow \begin{cases} 0.0003y_1 + y_2 = 2.0001 \\ y_1 = 1 \end{cases} \rightarrow y_2 = 2.0001 - 0.0003 = 1.9998$$

Step 2. Solve $Ux = y$ for x .

$$Ux = y \Rightarrow \begin{cases} x_1 + x_2 = 1 \\ 2.9997x_2 = 1.9998 \end{cases} \rightarrow x_2 = 0.6667$$

$$\boxed{x_1 = 0.3333}$$

$$P^T L = \begin{bmatrix} 3 & 1 & 0 \\ -1 & 2 & 1 \\ -2 & 0 & 0 \end{bmatrix}$$

$$P^T L y = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \rightarrow \begin{cases} 3y_1 + y_2 = 1 \rightarrow y_2 = 1 - 3(-3/2) = \frac{13}{2} \\ -y_1 + 2y_2 + y_3 = 2 \rightarrow y_3 = 2 - 3/2 - \frac{13}{2} \\ -2y_1 = 3 \rightarrow y_1 = -3/2 \end{cases}$$

$$O(\frac{2}{3}n^3)$$

Cholesky Factorization

Recall. A square matrix A is a symmetric matrix if $A^T = A$. In other words, $a_{ij} = a_{ji}$, for all i and j .

For a system with symmetric coefficient matrix, special solution techniques are available that offer computational advantages such as half the storage and half the computation time requirements.

One of the most popular approaches involves *Cholesky factorization*. In this factorization, a symmetric (positive definite) matrix A (with positive eigenvalues) can be decomposed, as in

is called a positive definite matrix if it has positive eigenvalues.

$$A = U^T U$$

where U is an upper triangular matrix. Let $U = (u_{ij})_{n \times n}$ is an upper triangular matrix.

Then, solving $A = U^T U$ for $\frac{n(n+1)}{2}$ unknowns u_{ij} leads the following solutions:

$$u_{11} = \sqrt{a_{11}}$$

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2} \quad i = 2, 3, \dots, n$$

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ii}} \quad j = i + 1, i + 2, \dots, n$$

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

$$A = U^T U = \begin{bmatrix} u_{11}^2 & & \\ & & \\ & & \end{bmatrix}$$

Example. Compute the Cholesky factorization for the symmetric matrix

$$A = \begin{pmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{pmatrix}$$

Note. Once the Cholesky factorization is obtained, the system $Ax = b$ can be solved according to the following steps:

1. Solve $U^T y = b$ for y
2. Solve $Ux = y$ for x

Note. MATLAB has a built-in function chol that generates the Cholesky factorization. It has the general syntax,

```
>> U = chol(X)
```

where U is an upper triangular matrix so that $U^T U = X$. Try it for the matrix A in the previous example. Note that by the function eig(A) make sure that the eigenvalues are positive.

LDLT
for symmetric
and not necessarily P.D.

Note. It should be noted that in the event that A is not square, a least-squares solution is obtained with an approach called QR factorization, which is not the scope of this course.

Matrix inverse using LU factorization

Recall. An $n \times n$ matrix A is said to be *nonsingular* (or *invertible*) if an $n \times n$ matrix A^{-1} exists with $AA^{-1} = A^{-1}A = I_n$. The matrix A^{-1} is called the inverse of A . A matrix without an inverse is called *singular* (or *noninvertible*).

Now, we will focus on how the inverse can be computed numerically. The inverse can be computed in a column-by-column fashion by generating solutions with standard unit vectors as the right-hand-side constants. Assume that A is an invertible $n \times n$ matrix and

$O(n^3)$ operations are needed to compute A^{-1} directly

$$e_i = \left(\underbrace{0, \dots, 0, 1, 0, \dots, 0}_{\text{the } i\text{-th comp. is one}} \right)^T \checkmark$$

e_i is the standard unit vector. Then, for $1 \leq i \leq n$, the unique solution of the following system of linear equations is the i -th column of A^{-1} .

$$A \times B = I \Rightarrow A \begin{bmatrix} B_1 & B_2 & B_3 & \dots & B_n \\ \uparrow & \uparrow & \uparrow & & \uparrow \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & e_3 & \dots & e_n \\ \uparrow & \uparrow & \uparrow & & \uparrow \end{bmatrix} \quad \begin{matrix} Ax = e_i. \\ \uparrow \end{matrix}$$

$$[AB_1 \quad AB_2 \quad AB_3 \quad \dots \quad AB_n] = [e_1 \quad e_2 \quad e_3 \quad \dots \quad e_n]$$

$$\begin{cases} AB_1 = e_1 \\ AB_2 = e_2 \\ \vdots \\ AB_n = e_n \end{cases}$$

n Systems of linear system of equations

The best way to implement such a calculation is with LU factorization. Recall that LU factorization provides a very efficient means to evaluate multiple right-hand-side vectors. Thus, it is ideal for evaluating the multiple unit vectors needed to compute the inverse.

Example. Use LU factorization to compute the inverse of the following matrix:

① Solve

$$A \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{ for the first column}$$

$$A = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}_{3 \times 3}$$

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

assuming that

② Solve

$$A \begin{pmatrix} b_{12} \\ b_{22} \\ b_{32} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ for the second column of } B$$

$$\checkmark L = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.1 & -0.02713 & 1 \end{bmatrix}, \quad \checkmark U = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.012 \end{bmatrix}$$

Solution.

Step 1. The first column of A^{-1} . Solving $Ld = e_1$ for the vector d implies that

$$d = (1 \quad -0.03333 \quad -0.1009)^T.$$

Thus, the first column of A^{-1} is obtained by solving $Ux = d$ for x which is

$$x = (0.33249 \quad -0.00518 \quad -0.01008)^T.$$

Step 2. The second column of A^{-1} . Like Step 1, solving $Ld = e_2$ for the vector d and the system $Ux = d$ for x implies that

$$x = (0.004944 \quad 0.142903 \quad 0.00271)^T.$$

Step 3. The third column of A^{-1} . Like Steps 1 and 2, solving $Ld = e_3$ for the vector d and the system $Ux = d$ for x implies that

$$x = (0.006798 \quad 0.004183 \quad 0.09988)^T.$$

Therefore:

$$A^{-1} = \begin{bmatrix} 0.33249 & 0.004944 & 0.006798 \\ -0.00518 & 0.142903 & 0.004183 \\ 0.01008 & 0.00271 & 0.09988 \end{bmatrix}$$

Example. Use LU factorization to compute the inverse of the following matrix:

$$A = \begin{bmatrix} 2 & 3 & -1 \\ 4 & 4 & -1 \\ -2 & -3 & 4 \end{bmatrix}_{3 \times 3}$$

Assuming that

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & 3 & -1 \\ 0 & -2 & 1 \\ 0 & 0 & 3 \end{bmatrix}.$$



Error Analysis and System Condition

The inverse matrix not only has applications in engineering and scientific computations, but also plays an important role to discern whether a system is ill-conditioned.

Three direct methods can be devised for ill-conditioning purpose:

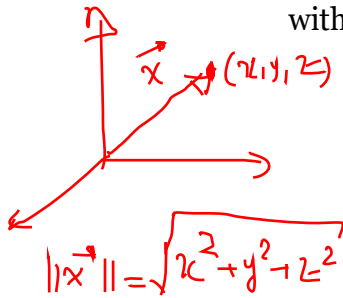
1. Scale the matrix of coefficients A so that the largest element in each row is 1. Invert the scaled matrix and if there are elements of A^{-1} that are several orders of magnitude greater than one, it is likely that the system is ill-conditioned.
2. Multiply the inverse by the original coefficient matrix and assess whether the result is close to the identity matrix. If not, it indicates ill-conditioning.
3. Invert the inverted matrix and assess whether the result is sufficiently close to the original coefficient matrix. If not, it again indicates that the system is ill-conditioned.

$$\|x\| = \left(\sum_{i=1}^n x_i^2 \right)^{1/2}$$

$$\|x\|_e = \|x\|_2$$

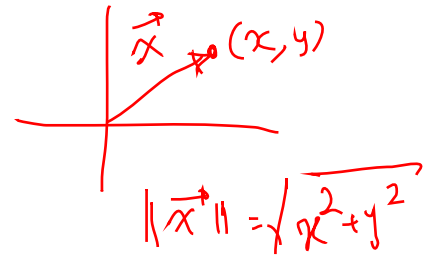
However, there would be a possibility of obtaining a single number to indicate ill-conditioning. This number is called the **condition number** which is based on the matrix norms.

Matrix and vector norm. A *vector norm* on \mathbb{R}^n is a function, $\|\cdot\|$, from \mathbb{R}^n into \mathbb{R} with the following properties:



$$\|\vec{x}\| = \sqrt{x^2 + y^2 + z^2}$$

- (i) $\|x\| \geq 0$ for all $x \in \mathbb{R}^n$,
- (ii) $\|x\| = 0$ if and only if $x = 0$,
- (iii) $\|\alpha x\| = |\alpha| \|x\|$ for all $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$,
- (iv) $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

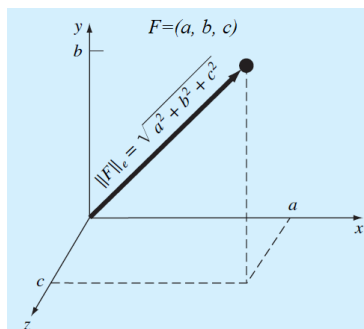


$$\|\vec{x}\| = \sqrt{x^2 + y^2}$$

Similarly, a matrix norm can be defined. Some popular matrix and vector norms are:

- Euclidean vector norm. For the vector $x = (x_1, x_2, \dots, x_n)^T$, it is defined by

$$\|x\| = \|x\|_2 = \|x\|_e = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} = \left(\sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}}$$



The concept can be extended further to a matrix $A_{n \times n}$, as in

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

$$\|A\|_F = (1 + 0 + 25 + 4)^{1/2} = 5.48$$

$$\|A\|_F = \left(\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2 \right)^{1/2}$$

which is given a special name—the Frobenius norm.

- Norm p or ℓ_p -norm. The Euclidean vector norm is a specific case of the p -norm, $1 \leq p < \infty$, which is defined by

$$x = \begin{pmatrix} -4 \\ 3 \\ 5 \\ -10 \end{pmatrix}$$

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

$$p=2 \rightarrow \|x\|_2 = \left(\sum x_i^2 \right)^{1/2}$$

$$p=1 \rightarrow \|x\|_1 = \sum |x_i|$$

$$p=\infty \rightarrow \|x\|_\infty$$

For $p = 1$, we have

$$\|x\|_1 = \sum_{i=1}^n |x_i|$$

$$\|x\|_1 = \sum |x_i| = 4 + 3 + 5 + 10 = 22$$

$$\|x\|_2 = \left(\sum x_i^2 \right)^{1/2} = 12.25$$

For $p = \infty$, the so-called uniform vector norm, we have

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

$$\|x\|_\infty = \max \{4, 3, 5, 10\} = 10$$

Using a similar approach, norms can be developed for matrices. For example,

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| \quad (\text{column-sum norm})$$

and

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \quad (\text{row-sum or uniform-matrix norm})$$

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 3 & 5 & 0 \\ -4 & -1 & 3 \end{pmatrix} \begin{array}{l} \rightarrow \|A'_1\| = 4 \\ \rightarrow \|A'_2\| = 8 \\ \rightarrow \|A'_3\| = 8 \end{array} \quad \begin{array}{l} \|A\|_1 = \max \{8, 8, 4\} = 8 \\ \|A\|_\infty = \max \{8, 8, 4\} = 8 \end{array}$$

13

$$\begin{array}{l} |A_1|_1 = 8 \\ |A_2|_1 = 8 \\ |A_3|_1 = 4 \end{array}$$

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

And

$$\|A\|_2 = (\mu_{\max})^{\frac{1}{2}} \quad (\text{spectral norm})$$

where μ_{\max} is the largest eigenvalue of $A^T A$.

$$c_1 \|x\|_2 \leq \|x\|_p \leq c_2 \|x\|_2$$

Matrix condition number. The condition number of the nonsingular matrix A relative to a norm $\|\cdot\|$ is

$$K(A) = \text{Cond}(A) = \|A\| \|A^{-1}\|$$

Technical lemma: The condition number is greater than or equal to one and it satisfies

$$\frac{\|\hat{x} - x^*\|}{\|x^*\|}$$

$$\frac{\|\Delta x\|}{\|x\|} \leq \text{Cond}(A) \frac{\|\Delta A\|}{\|A\|}$$

that is, the relative error of the norm of the computed solution can be as large as the relative error of the norm of the coefficients of A multiplied by the condition number.

Note. A matrix A is *well-conditioned* if $\text{Cond}(A)$ is close to 1 and is *ill-conditioned* when $\text{Cond}(A)$ is significantly greater than 1. Conditioning refers to the relative security that a small relative error of the norm of the coefficients of A implies a correspondingly accurate approximate solution.

Example. If the coefficients of A are known to t -digit precision (i.e., rounding errors are on the order of 10^{-t}) and $\text{Cond}(A) = 10^c$, the solution x may be valid to only $t - c$ digits (rounding errors $\approx 10^{c-t}$).

Example. The Hilbert matrix, which is notoriously ill-conditioned, can be represented generally as

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix}$$

Use the row-sum norm to estimate the matrix condition number for the 3×3 Hilbert matrix:

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

Solution. First, the matrix can be normalized so that the maximum element in each row is 1:

$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix} \xrightarrow{\text{use the adjoint method}} A^{-1} = \begin{bmatrix} 9 & -18 & 10 \\ -36 & 96 & -60 \\ 30 & -90 & 60 \end{bmatrix}$$

So,

$$\|A\|_{\infty} = \max\{1.833, 2.1667, 2.35\} = 2.35$$

$$\|A^{-1}\|_{\infty} = \max\{47, 192, 180\} = 192$$

$$\text{Cond}(A) = \|A\|_{\infty} \|A^{-1}\|_{\infty} = 2.35 \times 192 = 451.2$$

Remark. MATLAB has built-in functions to compute both norms and condition numbers:

```
>> norm(X,p)
```

and

```
>> cond(X,p)
```

where X is the vector or matrix and p designates the type of norm or condition number (1, 2, inf, or 'fro'). Note that the `cond` function is equivalent to

```
>> norm(X,p)*norm(inv(X),p)
```

Also, note that if p is omitted, it is automatically set to 2.

References

1. Chapra, Steven C. (2018). *Numerical Methods with MATLAB for Engineers and Scientists*, 4th Ed. McGraw Hill.
2. Burden, Richard L., Faires, J. Douglas (2011). *Numerical Analysis*, 9th Ed. Brooks/Cole Cengage Learning