Chapter 1: Linear Algebra and Systems of Linear Equations

1.1 BASICS OF LINEAR ALGEBRA

First, we introduce some basics of linear algebra, which will be used to describe and solve linear equations. We will just cover the very basics of it in this chapter. If you wish additional guidance, we suggest studying a linear algebra textbook.

1.1.1 Sets

We have discussed the set data structure `List, set and tupple` in Module 6 before. Here we look at it from a mathematics point of view. In mathematics, a **set** is a collection of objects. As defined earlier, sets are usually denoted by braces `{}`. For example, `S = {orange, apple, banana}` means `S` is the set containing "`orange`", "`apple`", and "`banana`."

The empty set is the set containing no objects and is typically denoted by empty braces such as `{}` or by ∅. Given two sets, A and B, the union of A and B is denoted by A ∪ B and is equal to the set containing all the elements of A and B. The intersection of A and B is denoted by A ∩ B and is equal to the set containing all the elements that belong to both A and B. In set notation, a colon is used to mean "such that." The usage of these terms will become apparent shortly. The symbol ∈ is used to denote that an object is contained in a set. For example, a ∈ A means "a is a member of A" or "a is in A." A backslash, \, in set notation means set minus. So, if a ∈ A then A\a means "A minus the element, a."

There are several standard sets related to numbers, for example, natural numbers, whole numbers, integers, rational numbers, irrational numbers, real numbers, and complex numbers. A description of each set and the symbol used to denote it is shown in the following table.

Set Name Symbol	Symbol	Description
Naturals	$\mathbb{N}$	$\mathbb{N} = \{1, 2, 3, 4,...\}$
Wholes	$\mathbb{W}$	$\mathbb{W} = \mathbb{N} \cup \{0\}$
Integers	$\mathbb{Z}$	$\mathbb{Z} = \mathbb{W} \cup \{-1,-2,-3,...\}$
Rationals	$\mathbb{Q}$	$\mathbb{Q} = \left\{\frac{p}{q} : p \in \mathbb{Z}, q \in \mathbb{Z} \setminus \{0\}\right\}$
Irrationals	$\mathbb{I}$	$\mathbb{I}$ is the set of real numbers not expressible as a fraction of integers
Reals	$\mathbb{R}$	$\mathbb{R} = \mathbb{Q} \cup \mathbb{I}$
Complex numbers	$\mathbb{C}$	$\mathbb{C} = \{a + bi : a, b \in \mathbb{R}, i = \sqrt{-1}\}$

Problem 1.1 Let *S* be the set of all real (x, y) pairs such that $x^2 + y^2 = 1$. Write *S* using set notation.

Ans:

1.1.2 Vectors

The set $\mathbb{R}^n$ is the set of all n-tuples of real numbers. In set notation, this is $\mathbb{R}^n = \{(x^1, x^2, x^3, \ldots, x^n) : x^1, x^2, x^3, \ldots, x^n \in \mathbb{R}\}$. For example, the set $\mathbb{R}^3$ represents the set of real triples, (x, y, z) coordinates, in 3D space.

A **vector** in Rn is an n-tuple, or point, in Rn. Vectors can be written horizontally (i.e., with the elements of the vector next to each other) in a **row vector**, or vertically (i.e., with the elements of the vector on top of each other) in a **column vector**. If the context of a vector is ambiguous, it usually means the vector is a column vector. The ith element of a vector, v, is denoted by v_i. The transpose of a column vector is a row vector of the same length, and the transpose of a row vector is a column vector. In mathematics, the transpose is denoted by a superscript T, or v^T. The **zero vector** is the vector in $\mathbb{R}^n$ containing all zeros.

The norm of a vector is a measure of its length. There are many ways of defining the length of a vector depending on the metric used (i.e., the distance formula chosen). The most common is called the L_2 norm, which is computed according to the distance formula. The L_2 **norm** of a vector v is denoted by $\|v\|_2$ and $\|v\|_2 = \sqrt{\sum_i v_i^2}$. This is sometimes also called Euclidean distance and refers to the "physical" length of a vector in 1, 2, or 3D space. The L_1 norm, or "Manhattan distance," is computed as $\|v\|_1 = \sum_i |v_i|$, and is named after the grid-like road structure in New York City. In general, the ***p*-norm**, L_p, of a vector is $\|v\|_p = \sqrt[p]{(\sum_i v_i^p)}$. The L_∞ **norm** is the *p*-norm, where $p = \infty$. The L_∞ norm is written as $\|v\|_\infty$ and is equal to the maximum absolute value in v

Problem1.2 Create a row vector and a column vector, and show their shape.

Ans:

Note! In Python, the row and column vectors can be tricky. As shown in the example above, to get the 1 row and 4 column or 4 row and 1 column vectors, we had to use a list of a list to specify it. You can define `np.array([1,2,3,4])`, but you will soon notice that it does not contain information about row or column.

vector → direction　magnitude
方向　大小

歐氏空間 ⇒ 三角形的內角和 = 180°

凹 <180°
凸 >180° ✓

(x,y,z) 坐標 $\vec{v} = a\vec{i} + b\vec{j} + c\vec{k}$

$= (a, b, c)$

$\vec{i}, \vec{j}, \vec{k}$ ⇒ Bases 基底 (正交 單範) → 長度為1
↳ 互相垂直

<Def> n維空間. 需有n個獨立的基底

If $\vec{a}_1, \vec{b}_1$ 互相垂直 $\Rightarrow \vec{a}_1 \cdot \vec{b}_1 = 0$.

$M^T \Rightarrow M_{m\times n} \Rightarrow M^T = M_{n\times m}$

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

Problem1.3 Transpose the row vector defined above into a column vector and calculate its L1, L2, and L∞ norm. Verify that the L∞ norm of a vector is equivalent to the maximum value of the elements in the vector.

Ans:

Vector addition is defined as the pairwise addition of the elements of the added vectors. For example, if v and w are vectors in $\mathbb{R}^n$, then $u = v + w$ is defined as having elements $u_i = v_i + w_i$.

Vector multiplication can be defined in several ways depending on the context. **Scalar multiplication** of a vector is the product of a vector and a **scalar** (i.e., a number in $\mathbb{R}$). Scalar multiplication is defined as the product of each element of the vector by the scalar. More specifically, if α is a scalar and v is a vector, then $u = \alpha v$ is defined as having elements $u_i = \alpha v_i$. Note that this is exactly how Python implements scalar multiplication with a vector.

Problem1.4 Show that $a(v + w) = av + aw$ (i.e., scalar multiplication of a vector distributes across vector addition). By vector addition, u = v + w is the vector with entries $u_i = v_i + w_i$.

Ans:

The **dot-product** of two vectors is the sum of the products of the respective elements and is denoted by ·, and $v \cdot w$ is read "v dot w." Therefore for $v, w \in \mathbb{R}^n$, $d = v \cdot w$ is defined as $d = \sum_{i=1}^{n} v_i w_i$. The **angle between two vectors**, θ, is defined by the formula:

$$v \cdot w = \|v\|_2 \|w\|_2 \cos\theta$$

The dot-product is a measure of how similarly directed the two vectors are. For example, the vectors (1, 1) and (2, 2) are parallel. If you compute the angle between them using the dot-product, you will find that $\theta = 0$. If the angle between the vectors is $\theta = \pi/2$, then the vectors are said to be perpendicular or **orthogonal**, and the dot-product is 0.

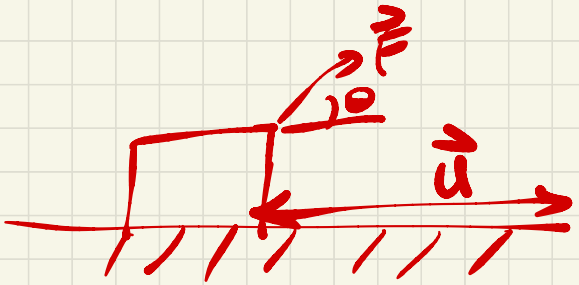
Problem1.5 Compute the angle between the vectors v = [10, 9, 3] and w = [2, 5, 12].

Ans:

<Def> 功

$\cos\theta$ → x方向的投影

$\sin\theta$ → y方向的投影



$$W = \vec{F}\cdot\vec{u}$$
$$= |\vec{F}||\vec{u}|\cdot\cos\theta$$

$\vec{F}$ 和 $\vec{u}$ 的夾角

$$\cos\theta = \frac{\vec{F}\cdot\vec{u}}{|\vec{F}||\vec{u}|} \Rightarrow \theta = \cos^{-1}\frac{\vec{F}\cdot\vec{u}}{|\vec{F}||\vec{u}|}$$

Finally, the cross-product between two vectors, v and w, is written as v × w. It is defined by $v \times w = \|v\|_2 \|w\|_2 \sin(\theta)$n, where θ is the angle between the v and w (which can be computed from the dot-product), and n is a vector perpendicular to both v and w with unit length (i.e., its length is one). The geometric interpretation of the cross-product is a vector perpendicular to both v and w, with the length equal to the area enclosed by the parallelogram created by the two vectors.

Problem1.6 Given the vectors v = [0, 2, 0] and w = [3, 0, 0], use the `NumPy` function `cross` to compute the cross-product of `v` and `w`.

Ans:

Assuming that S is a set in which addition and scalar multiplication are defined, a l**inear combination** of S is defined as

$$\sum \alpha_i s_i$$

where α_i is any real number, and s_i is the ith object in S. Sometimes the αi values are called the **coefficients** of s_i.

Linear combinations can be used to describe numerous things. For example, a grocery bill can be written $\sum c_i n_i$, where c_i is the cost of item i and ni is the number of items i purchased. Thus, the total cost is a linear combination of the items purchased.

A set is called **linearly independent** if no object in the set can be written as a linear combination of the other objects in the set. For the purposes of this book, we will only consider the linear independence of a set of vectors. A set of vectors that is not linearly independent is **linearly dependent**.

Problem1.7 Given the row vectors v = [0, 3, 2], w = [4, 1, 1], and u = [0,−2, 0], write the vector x = `[-8,-1, 4] as a linear combination of v, w, and u.`

Ans:

Problem1.8 Determine by inspection whether the following set of vectors is linearly independent: v = [1, 1, 0], w = [1, 0, 0], u = [0, 0, 1].

Ans:

<Def> linear combination

$c_1 a_1 + c_2 a_2 + \cdots + c_n a_n$

$\Rightarrow \sum c_i a_i$

tensor $\Rightarrow c_i a_i$

for every $c_1 a_1 + \cdots + c_n a_n = 0$

又 $c_1 = c_2 = c_3 \cdots c_n = 0$ 線性獨立

有之為線性相依

1.1.3 Matrices

An $m \times n$ **matrix** is a rectangular table of numbers consisting of m rows and n columns. The norm of a matrix can be considered as a particular kind of vector norm. If we treat the $m \times n$ elements of M as the elements of an mn-dimensional vector, then the p-norm of this vector can be written as

$$\|M\|_P = \sqrt[P]{\sum_i^m \sum_j^n |a_{ij}|^p}.$$

It is possible to calculate the matrix norm using the same `norm` function in `NumPy` as that for a vector.

Matrix addition and scalar multiplication for matrices work the same way as for vectors. However, **matrix multiplicatio**n between two matrices, P and Q, is defined when P is an $m \times p$ matrix and Q is a $p \times n$ matrix. The result of $M = P\,Q$ is a matrix M that is $m \times n$. The dimension p is called the **inner matrix dimension**, and the inner matrix dimensions must match (i.e., the number of columns in P and the number of rows in Q must be the same) for matrix multiplication to be defined. The dimensions m and n are called the **outer matrix dimensions**. Formally, if P is $m \times p$ and Q is $p \times n$, then $M = P\,Q$ is defined as

$$M_{ij} = \sum_{k=1}^{P} P_{ik} Q_{kj}$$

The product of two matrices P and Q in Python is achieved by using the **dot** method in `NumPy`. The **transpose** of a matrix is a reversal of its rows with its columns. The transpose is denoted by a superscript, T, such as M^T is the transpose of matrix M. In Python, the method `T` for a `NumPy` array is used to get the transpose. For example, if `M` is a matrix, then `M.T` is its transpose.

Problem1.9 Let matrices P and Q be [[1, 7], [2, 3], [5, 0]] and [[2, 6, 3, 1], [1, 2, 3, 4]], respectively. Com-pute the Python matrix product of `P` and `Q`. Show that the product of `Q` and `P` will produce an error.

Ans:

A **square matrix** is an $n \times n$ matrix, that is, it has the same number of rows as columns. The **determinant** is an important property of square matrices. It is a special number that can be calculated directly from a square matrix. The determinant is denoted by `det`, both in mathematics and in `NumPy`'s `linalg` package. Some examples of the use of determinant will be described later.

In the case of a 2×2 matrix, the determinant is

$$|M| = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

Similarly, in the case of a 3×3 matrix, the determinant is:

$$|M| = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = a\begin{bmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{bmatrix} - b\begin{bmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{bmatrix} + c\begin{bmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{bmatrix}$$

$$= a\begin{bmatrix} e & f \\ h & i \end{bmatrix} - b\begin{bmatrix} d & f \\ g & i \end{bmatrix} + c\begin{bmatrix} d & e \\ g & h \end{bmatrix}$$

$$= aei + bfg + cdh - ced - bdi - afh$$

We can use a similar approach to calculate the determinant for a higher-dimensional matrix, but it is much easier to calculate using Python. See the example below to calculate the determinant in Python.

The **identity matrix** is a square matrix with 1s on the diagonal and 0s elsewhere. The identity matrix is usually denoted by *I* and is analogous to the real number identity, 1. That is, multiplying any matrix by *I* (of compatible size) will produce the same matrix.

Problem1.10 Find the determinant of matrix *M* = [[0, 2, 1, 3], [3, 2, 8, 1], [1, 0, 0, 3], [0, 3, 2, 1]]. Use the `np.eye` function to produce a 4 × 4 identity matrix, *I*. Multiply *M* by *I* to show that the result is *M*.

Ans:

The **inverse** of a square matrix *M* is a matrix of the same size, N, such that $M \cdot N = I$. The inverse of a matrix is analogous to the inverse of a real number. For example, the inverse of 3 is 1 3 because $(3)(\frac{1}{3}) = 1$. A matrix is said to be **invertible** if it has an inverse. The inverse of a matrix is unique, that is, for an invertible matrix, there is only one inverse for that matrix. If *M* is a square matrix, its inverse is denoted by M^{-1} in mathematics, and it can be computed in Python using the function `inv` from `NumPy`'s `linalg` package.

For a 2 × 2 matrix, the analytical solution of the matrix inverse is

$$M^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{|M|}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}^{-1}$$

Calculating the matrix inverse for the analytical solution becomes complicated as the dimension of the matrix increases. There are many other methods which can make things easier, such as Gaussian elimination, Newton's method, eigendecomposition, etc. We will introduce some of these methods after we learn how to solve a system of linear equations (as the process is essentially the same).

Recall that zero has no inverse for multiplication in the setting of real numbers. Similarly, there are matrices that do not have inverses. These matrices are called **singular**. Matrices that do have an inverse are called **nonsingular**.

$M_{n\times n}$ 存在 $N_{n\times n}$

$\Rightarrow MN = I$ 单位矩阵

$N = M^{-1}$

One way to determine if a matrix is singular is by computing its determinant. If the determinant is 0, then the matrix is singular; if not, the matrix is non-singular.

Problem1.11 The matrix M (in the previous example) has a nonzero determinant. Compute the inverse of M. Show that the matrix P = [[0, 1, 0], [0, 0, 0], [1, 0, 1]] has a determinant value of zero, and therefore has no inverse.

Ans:

∵ det[P] = 0 ⇒ P^{-1} 不存在

A matrix that is close to being singular (i.e., the determinant is close to zero) is called **ill-conditioned**. Although ill-conditioned matrices have inverses, they are problematic numerically in the same way that dividing a number by a very, very small number is problematic. That is, it can result in computations that result in overflow, underflow, or numbers small enough to result in significant round-off errors. The **condition number** is a measure of how ill-conditioned a matrix is: it is defined as the norm of the matrix times the norm of the inverse of the matrix, that is, $\|M\|\,\|M\|^{-1}$. In Python, it can be computed using `NumPy`'s function `cond` from `linalg`. The higher the condition number, the closer the matrix to being singular.

The **rank** of an m × n matrix A is the number of linearly independent columns or rows of A and is denoted by rank(A). It can be shown that the number of linearly independent rows is always equal to the number of linearly independent columns for any matrix. A matrix has **full rank** if rank(A) = min(m, n). The matrix A is also of full rank if all of its columns are linearly independent. An **augmented matrix** is a matrix A concatenated with a vector y and is written [A, y]. This is commonly read as "A augmented with y." You can use `np.concatenate` to concatenate. If rank([A, y]) = rank(A) + 1, then the vector y is "new" information. That is, it cannot be created as a linear combination of the columns in A. Rank is an important characteristic of matrices because of its relationship to solutions of linear equations, which is discussed in the last section of this chapter.

Problem1.12 For the matrix A = [[1, 1, 0], [0, 1, 0], [1, 0, 1]], compute the condition number and rank. If y = [[1], [2], [1]], get the augmented matrix [A, y].

Ans:

1.2 LINEAR TRANSFORMATIONS

For any vectors x and y, and scalars a and b, we say that a function F is a **linear transformation** if

$$F\ (ax\ +\ by)\ =\ aF\ (x)\ +\ bF\ (y).$$

Problem1.13 Let x be a vector and let $F(x)$ be defined by $F\ (x)\ =\ Ax$, where A is a rectangular matrix of appropriate size. Show that $F(x)$ is a linear transformation.

Ans:

If A is an $m\ \times\ n$ matrix, then there are two important subspaces associated with A: one is $\mathbb{R}^n$, and the other is $\mathbb{R}^m$. The **domain** of A is a subspace of $\mathbb{R}^n$. It is the set of all vectors that can be multiplied by A on the right. The range of A is a subspace of m. It is the set of all vectors y such that $y\ =\ Ax$. It can be denoted as $R(A)$, where $R(A)\ =\ \{y\ \in\ Rm\ :\ Ax\ =\ y\}$. Another way to think about the range of A is as the set of all linear combinations of the columns in A, where xi is the coefficient of the ith column in A. The null space $N\ (A)\ =\ \{x\ \in\ Rn\ :\ Ax\ =\ 0m\}$ is the subset of vectors x in the domain of A such that $Ax = 0_m$, where 0m is the zero vector (i.e., a vector in $\mathbb{R}^m$ with all zeros).

Problem1.14 Let A = [[1, 0, 0], [0, 1, 0], [0, 0, 0]] and let the domain of A be $\mathbb{R}^3$. Characterize the range and nullspace of A.

Ans:

1.3 SYSTEMS OF LINEAR EQUATIONS

A linear equation is an equality of the form

$\sum_{i=1}^{n} a_i x_i = y$

where ai are scalars, x_i are unknown variables in $\mathbb{R}$, and y is a scalar.

Problem1.15 Determine which of the following equations is linear and which is not. For the ones that are not linear, can you manipulate them to make them linear?

$$3x_1 + 4x_2 - 3 = -5x_3,$$

$$\frac{-x_1+x_2}{x_3} = 2,$$

$$x_1x_2 + x_3 = 5.$$

Ans:

A **system of linear equations** is a set of linear equations that share the same variables. Consider the following system of linear equations:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n-1}x_{n-1} + a_{1,n}x_n &= y_1, \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n-1}x_{n-1} + a_{2,n}x_n &= y_2, \\ &\\ a_{m-1,1}x_1 + a_{m-1,2}x_2 + \cdots + a_{m-1,n-1}x_{n-1} + a_{m-1,n}x_n &= y_{1m-}, \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n-1}x_{n-1} + a_{m,n}x_n &= y_m, \end{aligned}$$

Where $a_{i,j}$ and y_i are real numbers. The **matrix form** of a system of linear equations is **Ax** = **y**, where A is an $m \times n$ matrix, $A(i,j) = a_{i,j}$, y is a vector in $\mathbb{R}^m$, and x is an unknown vector in $\mathbb{R}^n$. The matrix form is shown below:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{1,2} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

If you carry out the matrix multiplication, you will see that you arrive back at the original system of equations.

$$Ax = y$$

Problem1.16 Put the following system of equations into matrix form:

$$\left.\begin{aligned} 4x + 3y - 5z &= 2, \\ -2x - 4y + 5z &= 5, \\ 7x + 8y &= -3, \\ x + 2z &= 1, \\ 9 + y - 6z &= 6, \end{aligned}\right\}$$

Ans:

$$\begin{bmatrix} 4 & 3 & -5 \\ -2 & -4 & 5 \\ 7 & 8 & 0 \\ 1 & 0 & 2 \\ 9 & 1 & -6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ -3 \\ 1 \\ 6 \end{bmatrix}$$

1.4 SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Consider a system of linear equations in matrix form, $Ax = y$, where A is an $m \times n$ matrix. Recall that this means there are m equations and n unknowns in our system. A **solution** to this system of linear equations is an x in Rn that satisfies the matrix form equation. Depending on the values that populate A and y, there are three distinct possible solutions for x. Either there is no solution for x, or there is one unique solution for x, or there are infinitely many solutions for x. This fact is not shown in this text.

Case 1: There is no solution for x. If rank($[A, y]$) = rank(A) + 1 then y is linearly independent from the columns of A. Therefore, because y is not in the range of A, by definition there cannot be an x that satisfies the equation. Thus, comparing rank($[A, y]$) and rank(A) provides an easy way to check if there are no solutions to a system of linear equations.

Case 2: There is a unique solution for x. If rank($[A, y]$) = rank(A), then y can be written as a linear combination of the columns of A, and there is at least one solution for the matrix equation. For there to be only one solution, rank(A) = n must also be true. In other words, the number of equations must be exactly equal to the number of unknowns. To see why this property results in a unique solution, consider the following three relationships between m and n : $m < n, m = n$, and $m > n$.

- For the case $m < n$ rank(A) = n cannot possibly be true because this means we have a "fat" matrix with fewer equations than unknowns. Thus, we do not need to consider this subcase.
- When m = n and rank(A) = n, A is square and invertible. Since the inverse of a matrix is unique, the matrix equation $Ax = y$ can be solved by multiplying each side of the equation on the left by A^{-1}. This results in $A^{-1}Ax = A^{-1}y \rightarrow Ix = A^{-1}y \rightarrow x = A^{-1}y$, which gives the unique solution to the equation.
- If m>n, then there are more equations than unknowns; however, if rank(A) = n, then it is possible to choose n equations (i.e., rows of A) such that if these equations are satisfied, then the remaining $m - n$ equations will also be satisfied. In other words, they are redundant. If the m − n redundant equations are removed from the system, then the resulting system has an A matrix that is $n \times n$ and invertible. These facts are not proven in this text. The new system then has a unique solution, which is valid for the whole system.

$Ax = y$

$x = A^{-1}y$ direct method.

Case 3: There are infinitely many solutions for x. If rank($[A, y]$) = rank(A), then y is in the range of A, and there is at least one solution for the matrix equation; however, if rank(A) < n, then there are infinitely many solutions. Although it is not shown here, if rank(A) < n, then there is at least one nonzero vector, n, that is in the null space of A (Actually, there are infinitely many null space vectors under these conditions.). If n is in the nullspace of A, then $An = 0$ by definition. Now if x^* is a solution to the matrix equation $Ax = y$, then, necessarily, $Ax^* = y$; however, $Ax^* + An = y$ or $A(x^* + n) = y$. Therefore, x* + n is also a solution for $Ax = y$. In fact, since A is a linear transformation, $x^* + \alpha n$ is a solution for any real number α (you should try to show this on your own). Since there are infinitely many acceptable values for α, there are infinitely many solutions for the matrix equation.

The rest of the chapter will discuss how to solve a system of equations which has a unique solution. First, we will discuss some of the common methods that you will most likely come across in your work and then we will show you how to solve systems in Python.

Let us say we have n equations with n variables, Ax = y, as follows:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{1,2} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

1.4.1 GAUSS ELIMINATION METHOD

The Gauss elimination method is a procedure that turns the matrix A into an upper-triangular form to solve the system of equations. Let us use a system of four equations and four variables to illustrate the idea. Gauss elimination essentially turns the system of equations into

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} \\ 0 & 0 & a'_{3,3} & a'_{3,4} \\ 0 & 0 & 0 & a'_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

By returning to the matrix form using this method, we can see the equations turn into:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 &= y_1, \\ a'_{2,2}x_2 + a'_{2,3}x_3 + a'_{2,4}x_4 &= y'_2, \\ a'_{3,3}x_3 + a'_{3,4}x_4 &= y'_3, \\ a'_{4,4}x_4 &= y'_4. \end{aligned}$$

Now, x_4 can be easily solved for by dividing both sides by $a'_{4,4}$, and then by substituting the result into the third equation to solve for x_3. With x_3 and x_4, we can substitute them into the second equation to solve for x_3, and we are now able to solve for all x. We solved the system of equations

bottom-up; this is called **backward substitution**. Note that, if A were a lower-triangular matrix, we would solve the system top-down by **forward substitution**.

To illustrate how we solve the equations using Gauss elimination, we use the example below.

Problem1.17 Use Gauss elimination to solve the following equations:

$$\begin{aligned} 4x_1 + 3x_2 - 5x_3 &= 2, \\ -2x_1 - 4x_2 + 5x_3 &= 5, \\ 8x_1 + 8x_2 &= -3, \end{aligned}$$

Ans:

$$\begin{bmatrix} 4 & 3 & -5 \\ -2 & -4 & 5 \\ 8 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ -3 \end{bmatrix}$$

Step 1. 계수 extend form

$$\left[\begin{array}{ccc|c} 4 & 3 & -5 & 2 \\ -2 & -4 & 5 & 5 \\ 8 & 8 & 0 & -3 \end{array}\right] \div 4 \Rightarrow \left[\begin{array}{ccc|c} 1 & \frac{3}{4} & -\frac{5}{4} & \frac{1}{2} \\ -2 & -4 & 5 & 5 \\ 8 & 8 & 0 & -3 \end{array}\right] \begin{matrix} \\ \times 2 \\ \times 8 \end{matrix}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & \frac{3}{4} & -\frac{5}{4} & \frac{1}{2} \\ 0 & -\frac{5}{2} & \frac{15}{4} & 6 \\ 0 & 2 & 10 & -7 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} 1 & \frac{3}{4} & -\frac{5}{4} & \frac{1}{2} \\ 0 & 1 & -1 & -11/5 \\ 0 & 0 & 12 & -1\frac{1}{5} \end{array}\right]$$

Note! Sometimes the first element in the first row is zero. When this is the case, switch the first row with a nonzero first element row, then follow the same procedure as outlined above. We are using the "pivoting" Gauss elimination method here. Note that there is also a "naive" Gauss elimination method which assumes that the pivot values will never be zero.

1.4.2 GAUSS–JORDAN ELIMINATION METHOD

Gauss–Jordan elimination solves systems of equations. It is a procedure to turn A into a diagonal form such that the matrix form of the equations becomes

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1^t \\ y_2^t \\ y_3^t \\ y_4^t \end{bmatrix}$$

Essentially, the equations become:

$$x_1 + 0 + 0 + 0 = y_1^t,$$

$$0 + x_2 + 0 + 0 = y_2^t,$$

$$0 + 0 + x_3 + 0 = y_3^t,$$

$$0 + 0 + 0 + x_4 = y_4^t,$$

Let us solve another equation system by using the above example as a blueprint.

Problem1.18 Use Gauss–Jordan elimination to solve the following equations:

$4x_1 + 3x_2 - 5x_3 = 2,$

$-2x_1 - 4x_2 + 5x_3 = 5,$

$8x_1 + 8x_2 = -3.$

Ans:

1.4.3 LU DECOMPOSITION METHOD

The two methods shown above involve changing both A and y at the same time while trying to turn A to an upper triangular or diagonal matrix form. Sometimes we may have same set of equations but different sets of y for different experiments. This is actually quite common in the real world, where we have different experiment observations y_a, y_b, y_c,…Therefore, we must solve $Ax = y_a$, $Ax = y_b$…

many times, since every time the [A, y] will change. Obviously, this is really inefficient. Is there a method by which we only change the left side of A but not the right-hand side y?

The LU decomposition method changes the matrix A only, instead of y. It is ideal for solving the system with the same coefficient matrices A but different constant vectors y. The LU decomposition method aims to turn A into the product of two matrices L and U, where L is a lower triangular matrix while U is an upper triangular matrix. With this decomposition, we convert the system of equations to the following form

$$LU_x = y \rightarrow \begin{bmatrix} l_{1.1} & 0 & 0 & 0 \\ l_{2.1} & l_{2.2} & 0 & 0 \\ l_{3.1} & l_{3.2} & l_{3.3} & 0 \\ l_{4.1} & l_{4.2} & l_{4.3} & l_{4.4} \end{bmatrix} \begin{bmatrix} u_{1.1} & u_{1.2} & u_{1.3} & u_{1.4} \\ 0 & u_{2.2} & u_{2.3} & u_{2.4} \\ 0 & 0 & u_{3.3} & u_{3.4} \\ 0 & 0 & 0 & u_{4.4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

If we define $U_x = M$, then the above equations become:

$$\begin{bmatrix} l_{1.1} & 0 & 0 & 0 \\ l_{2.1} & l_{2.2} & 0 & 0 \\ l_{3.1} & l_{3.2} & l_{3.3} & 0 \\ l_{4.1} & l_{4.2} & l_{4.3} & l_{4.4} \end{bmatrix} M = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

We can easily solve the above problem by forward substitution (the opposite of the backward substitution as we saw in Gauss elimination method). After we solve for M, we can easily solve the rest of the problem using backward substitution:

$$\begin{bmatrix} u_{1.1} & u_{1.2} & u_{1.3} & u_{1.4} \\ 0 & u_{2.2} & u_{2.3} & u_{2.4} \\ 0 & 0 & u_{3.3} & u_{3.4} \\ 0 & 0 & 0 & u_{4.4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix}.$$

But how do we obtain the L and U matrices? There are different ways to obtain the LU decomposition. Below is one example that uses the Gauss elimination method. From the above, we know that we obtain an upper triangular matrix after we conduct the Gauss elimination. At the same time, we also obtain the lower triangular matrix even though it is never explicitly written out. During the Gauss elimination procedure, the matrix A turns into the product of two matrices as shown below. The right upper triangular matrix is the one we obtained earlier. The diagonal elements in the left lower triangular matrix are 1, and the elements below the diagonal elements are the multipliers that

multiply the pivot equations to eliminate the elements during the calculation:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{2.1} & 1 & 0 & 0 \\ m_{3.1} & m_{3.2} & 1 & 0 \\ m_{4.1} & m_{4.2} & m_{4.3} & 1 \end{bmatrix} \begin{bmatrix} u_{1.1} & u_{1.2} & u_{1.3} & u_{1.4} \\ 0 & u_{2.2} & u_{2.3} & u_{2.4} \\ 0 & 0 & u_{3.3} & u_{3.4} \\ 0 & 0 & 0 & u_{4.4} \end{bmatrix}.$$

Note that we obtain both L and U at the same time when we perform the Gauss elimination. Using the above example, where U is the one we used before to solve the equations, and L is composed of the multipliers (you can check the examples in the Gauss elimination section), we obtain:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 2 & -0.8 & 1 \end{bmatrix},$$

$$\text{U} = \begin{bmatrix} 4 & 3 & -5 \\ 0 & -2.5 & 2.5 \\ 0 & 0 & 60 \end{bmatrix}.$$

Problem1.19 Verify that the above L and U matrices are the LU decomposition of matrix A. The result should be A = LU.

Ans:

1.4.4 ITERATIVE METHODS – GAUSS–SEIDEL METHOD

The methods introduced above are all direct methods where the solution is computed using a finite number of operations. This section introduces a different class of methods, namely the iterative methods, or indirect methods. They start with an initial guess of the solution and then repeatedly improve the solution until the change of the solution is below a chosen threshold. In order to use this iterative process, we first need to write the explicit form of a system of equations. If we have a system of linear equations

$$\begin{bmatrix} a_{1.1} & a_{1.2} & \cdots & a_{1.n} \\ a_{2.1} & a_{2.2} & \cdots & a_{2.n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m.1} & a_{m.2} & \cdots & a_{m.n} \end{bmatrix} \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{vmatrix},$$

we can write its explicit form as

$$x_i = \frac{1}{a_{i,i}} \left[y_i - \sum_{j=1, j\neq 1}^{j=n} a_{i,j} x_j \right].$$

This is the basics of the iterative methods; we can assume initial values for all the x, and use it as $x^{(0)}$. In the first iteration, we can substitute $x^{(0)}$into the right-hand side of the explicit equation above to obtain the first iteration solution $x^{(1)}$. By substituting $x^{(1)}$into the equation, we obtain $x^{(2)}$, and the iterations continue until the difference between x(k) and $x^{(k-1)}$ is smaller than some predefined value.

Iterative methods require having specific conditions for the solution to converge. A sufficient, but not necessary, condition of the convergence is that the coefficient matrix a is **diagonally dominant.** This means that in each row of the matrix of coefficients a, the absolute value of the diagonal element is greater than the sum of the absolute values of the off-diagonal elements. If the coefficient matrix satisfies this condition, the iterations will converge to the solution. Note that the solution process might still converge even when this condition is not satisfied.

1.4.4.1 Gauss–Seidel Method

The **Gauss–Seidel method** is a specific iterative method that is always using the latest estimated value for each element in x. For example, first assume that the initial values for x_2, x_3,..., x_n (except for x_1) are given and calculate x_1. Using the calculated x1 and the rest of the x (except for x_2), we can calculate x_2. Continuing in the same manner and calculating all the elements in *x* will conclude the first iteration. The unique part of the Gauss–Seidel method is the use of the latest value to calculate the next value in *x* . Such iterations are continued until the value converges. Let us use this method to solve the same problem we just solved above.

EXAMPLE: Solve the following system of linear equations using Gauss–Seidel method using a predefined threshold $\in = 0.01$. Remember to check if the converge condition is satisfied or not.

$$8x_1 + 3x_2 - 3x_3 = 14,$$
$$-2x_1 - 28x_2 + 5x_3 = 5,$$
$$83_1 + 35 + 10x_3 = -8,$$

Let us first check if the coefficient matrix is diagonally dominant or not.

1.4.5 SOLVING SYSTEMS OF LINEAR EQUATIONS IN PYTHON

The examples presented above demonstrated the various methods you can use to solve systems of linear equations. This is also very easy to do in Python, as shown below. The easiest way to get a solution is via the solve function in `NumPy`.

Problem1.20 Use `numpy.linalg.solve` to solve the following equations:

$4x_1 + 3x_2 - 5x_3 = 2,$

$-2x_1 - 4x_2 + 5x_3 = 5,$

$8x_1 + 8x_2 = -3.$

Ans:

We get the same results as those in the previous section when calculated by hand. Under the "hood," the solver is actually doing an LU decomposition to get the results. If you can check the help of the function, you will see it needs the input matrix to be square and of full rank, i.e., all rows (or, equivalently, columns) must be linearly independent.

Problem1.21 Try to solve the above equations using the matrix inversion approach.

Ans:

We can also obtain the L and U matrices used in the LU decomposition using the SciPypackage.

Problem1.22 Get the L and U for the above matrix A.

Ans:

Why do we obtain different L and U from those calculated by hand in the last section? You will also see that there is a **permutation matrix** P that is returned by the lu function. This permutation matrix records how it changes the order of the equations for easier calculation purposes. For example, if the first element in the first row is zero, it cannot be the pivot equation since you cannot turn the first elements in other rows to zero; therefore, we need to switch the order of the equations to get a new pivot equation. If you multiply P and A, you will see that this permutation matrix reverses the order of the equations for this case.

Problem1.23 Multiply P and A and see what is the effect of the permutation matrix on A.

Ans:

1.4.6 MATRIX INVERSION

We defined the inverse of a square matrix M as a matrix of the same size, M^{-1}, such that $M \cdot M^{-1} = M^{-1} \cdot M = I$. If the dimension of the matrix is high, the analytical solution for the matrix inversion will be complicated. Therefore, we need some other efficient ways to obtain the inverse of the matrix.

Let us use a 4×4 matrix for illustration. Suppose we have

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix},$$

Therefore, we will have:

$$M \cdot X = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We can rewrite the above equation as four separate equations, i.e.,

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \\ x_{4,1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_{1,2} \\ x_{2,2} \\ x_{3,2} \\ x_{4,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_{1,3} \\ x_{2,3} \\ x_{3,3} \\ x_{4,3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_{1,4} \\ x_{2,4} \\ x_{3,4} \\ x_{4,4} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Solving the above four systems of equations will provide the inverse of the matrix. We can use any method introduced previously to solve these equations (such as Gauss elimination, Gauss–Jordan, and LU decomposition). Below is an example of matrix inversion using the Gauss–Jordan method. Recall that in the Gauss–Jordan method, we convert our problem from

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

to

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ y_4 \end{bmatrix}$$

to obtain the solution. Essentially, we are converting

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & y_1 \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} & y_2 \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & y_3 \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & y_4 \end{bmatrix}$$

to

$$\begin{bmatrix} 1 & 0 & 0 & 0 & y_1' \\ 0 & 1 & 0 & 0 & y_2' \\ 0 & 0 & 1 & 0 & y_3' \\ 0 & 0 & 0 & 1 & y_4' \end{bmatrix}.$$

In summary, all we need to do is to convert

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & 1 & 0 & 0 & 0 \\ m_{2.1} & m_{2,2} & m_{2,3} & m_{2,4} & 0 & 1 & 0 & 0 \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & 0 & 0 & 1 & 0 \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & 0 & 0 & 0 & 1 \end{bmatrix}$$

to

$$\begin{bmatrix} 1 & 0 & 0 & 0 & m'_{1,1} & m'_{1,2} & m'_{1,3} & m'_{1,4} \\ 0 & 1 & 0 & 0 & m'_{2,1} & m'_{2,2} & m'_{2,3} & m'_{2,4} \\ 0 & 0 & 1 & 0 & m'_{3,1} & m'_{3,2} & m'_{3,3} & m'_{3,4} \\ 0 & 0 & 0 & 1 & m'_{4,1} & m'_{4,2} & m'_{4,3} & m'_{4,4} \end{bmatrix}.$$

Then the matrix

$$\begin{bmatrix} m'_{1,1} & m'_{1,2} & m'_{1,3} & m'_{1,4} \\ m'_{2,1} & m'_{2,2} & m'_{2,3} & m'_{2,4} \\ m'_{3,1} & m'_{3,2} & m'_{3,3} & m'_{3,4} \\ m'_{4,1} & m'_{4,2} & m'_{4,3} & m'_{4,4} \end{bmatrix}$$

is the inverse of M we are looking for

1.4.7 SUMMARY

What we have learned in this chapter

- Linear algebra is the foundation of many engineering fields.
- Vectors can be considered as points in $\mathbf{R}^n$; addition and multiplication are defined, although this is not necessarily the case for scalars.
- A set of vectors is linearly independent if none of the vectors can be written as a linear combination of the others.
- Matrices are tables of numbers. They have several important characteristics including the determinant, rank, and inverse.
- A system of linear equations can be represented by the matrix equation Ax = y.
- The number of solutions to a system of linear equations is related to rank(A) and rank([A, y]). It can be zero, one, or infinity.
- We can solve the equations using Gauss elimination, Gauss–Jordan elimination, LU decomposition, and the Gauss–Seidel method.
- We introduced methods to find the matrix inverse.