



CONTENTS

1	Lines and Planes in Three Dimensions	3
1.1	Points and Vectors in Space	3
1.2	Lines in Space	3
1.3	Planes	3
1.4	Distance from a Point to a Plane	4
1.5	Where Lines and Planes Meet	4
2	Determinants and Inverse Matrices	7
2.1	Determinants	7
2.2	Determinants in Python	11
2.3	Inverse Matrices	13
2.3.1	Finding an Inverse Matrix	15
2.3.2	Relation to $A\vec{x} = \vec{b}$	17
2.3.3	Matrix Inverse using NumPy	18
2.4	Summary	20
2.5	Where to Learn More	20
3	Matrices as Transformations	21
3.0.1	The Identity Transformation	22
3.1	Scaling and Shearing	24
3.2	Reflections	29
3.2.1	Reflection across the x-axis	29
3.2.2	Reflection across the y-axis	31
3.2.3	Reflection across the lines $y = x$ and $y = -x$	33

3.3	Rotations	36
3.4	What if I want to undo my transform?	38
3.5	Summary and looking ahead	40
4	Eigenvectors and Eigenvalues	43
4.1	Definition	44
4.2	Finding Eigenvalues and Eigenvectors	44
4.3	Example	45
4.4	Eigenbasis and Eigenspace	47
4.5	Eigenvalues and Eigenvectors in Python	49
4.6	Connection to transformations	49
4.7	Summary	50
4.8	Where to Learn More	51
5	Projections	53
5.1	Projections in Python	57
5.2	Where to Learn More	57
6	The Gram-Schmidt Process	59
6.1	The Process	59
6.2	Example Calculation	60
6.3	The Gram-Schmidt Process in Python	62
6.4	Where to Learn More	63
7	Singular Value Decomposition	65
7.1	Definition	65
7.2	Applications of SVD	65
7.3	Calculating SVD Manually	66
7.4	Singular Value Decomposition with Python	69
7.5	Sign Ambiguity	70
7.6	SVD Applied to Image Compression	71
7.7	Where to Learn More	72
A	Answers to Exercises	73
	Index	83

Lines and Planes in Three Dimensions

1.1 Points and Vectors in Space

Points in three dimensions are written (x, y, z) . A vector in space has three components, $\vec{v} = \langle a, b, c \rangle$. A vector gives a direction and a distance, and it can move a point to a new point.

Think about a classroom. A wall is a flat plane, and the line where two walls meet is a line in space. To describe a line or a plane with numbers, we need one point and a direction. The rest of this chapter shows how.

1.2 Lines in Space

A line in space is determined by one point $P_0 = (x_0, y_0, z_0)$ on it and a *direction vector* $\vec{d} = \langle a, b, c \rangle$ parallel to it. Every point on the line can be reached from P_0 by moving some multiple t of $\vec{d}$. This gives the *vector equation* of the line:

$$\vec{r}(t) = \langle x_0, y_0, z_0 \rangle + t\langle a, b, c \rangle$$

Writing the components separately gives the *parametric equations*:

$$x = x_0 + at, \quad y = y_0 + bt, \quad z = z_0 + ct$$

For example, the line through $(1, 2, 3)$ with direction $\langle 2, -1, 4 \rangle$ is $x = 1 + 2t$, $y = 2 - t$, $z = 3 + 4t$. When $t = 1$, the point is $(3, 1, 7)$.

1.3 Planes

A plane is determined by one point P_0 on it and a *normal vector* $\vec{n} = \langle a, b, c \rangle$, which is perpendicular to every direction in the plane. A point $P = (x, y, z)$ is on the plane exactly when the vector from P_0 to P is perpendicular to $\vec{n}$:

$$\vec{n} \cdot \langle x - x_0, y - y_0, z - z_0 \rangle = 0$$

Expanding this gives the linear equation

$$ax + by + cz = d, \quad \text{where } d = ax_0 + by_0 + cz_0$$

For the plane through $(1, 1, 1)$ with normal $\langle 2, -1, 2 \rangle$, we get $d = 2 - 1 + 2 = 3$, so the plane is $2x - y + 2z = 3$.

This connects to the last chapter: a plane through the origin is the span of two independent vectors lying in it. If you know two such vectors $\vec{u}$ and $\vec{v}$ instead of a normal, the cross product $\vec{u} \times \vec{v}$ gives you a normal.

1.4 Distance from a Point to a Plane

The distance from a point $P_0 = (x_0, y_0, z_0)$ to the plane $ax + by + cz = d$ is

$$D = \frac{|ax_0 + by_0 + cz_0 - d|}{\sqrt{a^2 + b^2 + c^2}}$$

The numerator measures how far P_0 is from satisfying the plane's equation, and the denominator scales that by the length of the normal vector.

For the origin and the plane $2x - y + 2z = 3$: the numerator is $|0 - 3| = 3$, and the denominator is $\sqrt{4 + 1 + 4} = 3$. The distance is 1.

1.5 Where Lines and Planes Meet

A line and a plane usually meet at one point. To find it, put the parametric equations of the line into the equation of the plane, and solve for t . Then substitute that t back into the line.

If the t 's cancel, the line is parallel to the plane. You end up with either a false statement like $1 = 5$ (the line never touches the plane) or a true one like $5 = 5$ (the line lies in the plane).

Worked example: Meeting point. Find where the line $x = 1 + 2t$, $y = 2 - t$, $z = 3 + 4t$ meets the plane $x + y + z = 11$.

Solution: Substitute the parametric equations into the plane:

$$(1 + 2t) + (2 - t) + (3 + 4t) = 11$$

This simplifies to $6 + 5t = 11$, so $t = 1$. Substituting $t = 1$ into the line gives the point $(3, 1, 7)$.

Exercise 1 A point on a line

Find the point on the line $x = 1 + 2t$,
 $y = 2 - t$, $z = 3 + 4t$ when $t = -1$.

Working Space

Answer on Page 73

Exercise 2 Equation of a plane

Find the equation of the plane through
 $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$.

Working Space

Answer on Page 73

Exercise 3 Distance to a plane

Find the distance from the origin to the
plane $x + y + z = 1$.

Working Space

Answer on Page 73

Exercise 4 Line meets a plane

Find where the line $x = 2 + t$, $y = -1 + 3t$,
 $z = 4 - t$ meets the plane $x + y + z = 8$.

Working Space

Answer on Page 73

Exercise 5 A line that misses

Show that the line $x = t$, $y = 1 + t$, $z = -2t$ never meets the plane $x + y + z = 5$.

Working Space

Answer on Page 73

Determinants and Inverse Matrices

2.1 Determinants

Every square matrix has a single number hiding inside it that answers a big question: are the columns independent? That number is the *determinant*. Put your vectors into the columns of a matrix, compute the determinant, and if it is zero, the vectors are dependent; if not, they are independent.

The determinant also has a beautiful geometric meaning. For a 2×2 matrix, it is the area of the parallelogram formed by the two column vectors (with a sign: negative if the columns are “flipped”). For a 3×3 matrix, it is the volume of the parallelepiped (a slanted box) formed by the three columns.¹

Think about a rectangle on graph paper. Its area is base times height. The determinant gives the area of a parallelogram in the same way. Let us plot the parallelogram for this matrix (see figure 2.1):

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

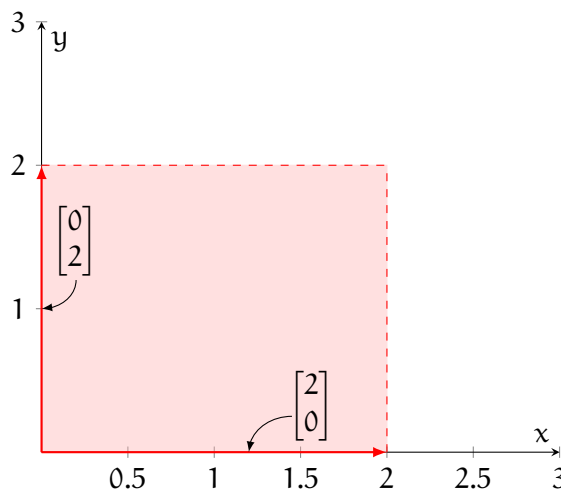


Figure 2.1: A parallelogram constructed from vectors $[2, 0]$ and $[0, 2]$

¹Note that determinants can only be found for square, $n \times n$ matrices.

2 by 2 Determinant

The formal definition for calculating the determinant of a 2 by 2 matrix A is:

$$\det(A) = (a \cdot d) - (b \cdot c)$$

where

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

For the matrix plotted above, the determinant is $(2 * 2) - (0 * 0)$. You can also see that 4.0 is the area, base (2) times height (2).

You can use the determinant to see what happens to a shape when it goes through a linear transformation. Let us scale the 2 by 2 matrix by 4:

$$\begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}$$

Plot it (see figure 2.2):

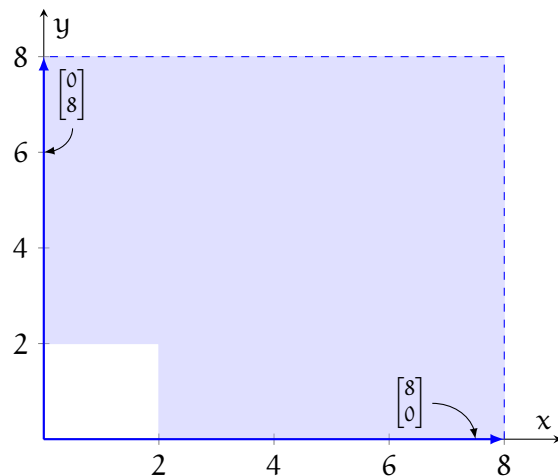


Figure 2.2: Scaling the matrix also scales the parallelogram.

Find the determinant using $(8 * 8) - (0 * 0) = 64$

You can see that scaling the matrix scaled the area by the scaling factor squared (see figure 2.3).

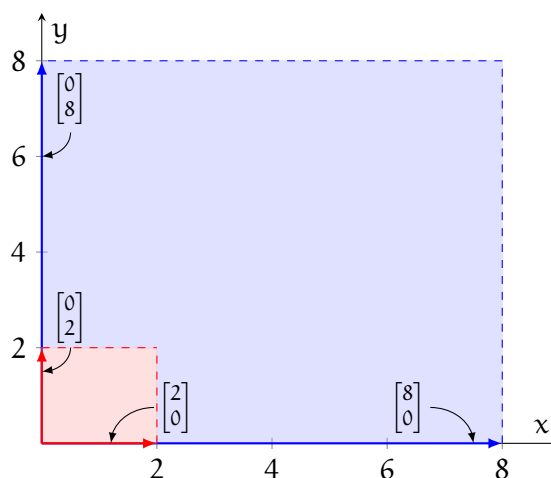


Figure 2.3: Scaling a matrix by a constant c increases the area of the parallelogram by a factor of c^2 .

We can show why this is true mathematically. Suppose we have a 2 by 2 matrix A :

$$A = \begin{bmatrix} w & x \\ y & z \end{bmatrix}$$

Then $\det(A) = wz - xy$. We can scale this matrix by a constant, c :

$$cA = c \cdot \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} cw & cx \\ cy & cz \end{bmatrix}$$

And we can take the determinant:

$$\det(cA) = \det \left(\begin{bmatrix} cw & cx \\ cy & cz \end{bmatrix} \right) = cw(cz) - cx(cy) = c^2(wz - xy) = c^2 \cdot \det(A)$$

Therefore, scaling a 2 by 2 matrix by a factor changes the determinant by that factor squared. What about higher dimensions? If each side of a cube were scaled by a factor of c , then the volume of the cube would change by a factor of c^3 (feel free to confirm this yourself). And if a tesseract (a four-dimensional cube) had each side scaled by a factor of c , then the hypervolume (four-dimensional volume) would be scaled by a factor of c^4 . Do you notice a pattern?

In fact, scaling an $n \times n$ matrix by a constant factor, c , changes the determinant of that $n \times n$ matrix by a factor of c^n .

What happens if the columns of a matrix are not independent? Let us plot this matrix

(see figure 2.4):

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

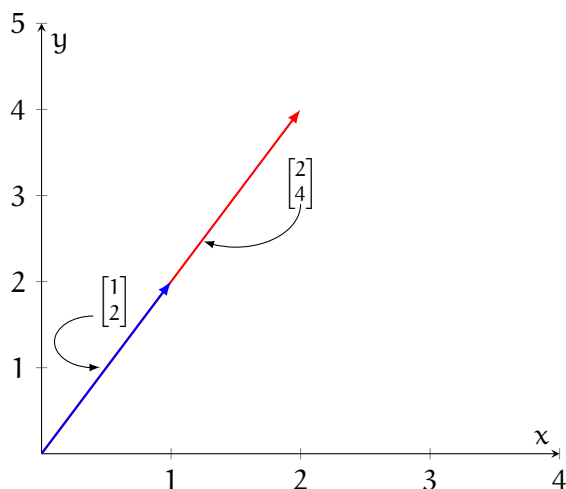


Figure 2.4: The vectors $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$ are co-linear, so there is no area between them and the determinant of $\begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix}$ is zero.

One vector overwrites the other. As you can see, the area is 0 because there is no space between the vectors. Therefore, the columns of the matrix are linearly dependent.

Exercise 6 Finding the Determinant

Plot the parallelogram represented by the columns of the matrix. What is the area of this parallelogram?

Working Space

1. $\begin{bmatrix} 1 & 4 \\ -3 & 1 \end{bmatrix}$

2. $\begin{bmatrix} 5 & -5 \\ 5 & -1 \end{bmatrix}$

3. $\begin{bmatrix} 0 & -5 \\ -2 & 0 \end{bmatrix}$

Answer on Page 73

Calculating the determinant for a 2 by 2 matrix is easy. For a larger matrix, finding the determinant is more complex and requires breaking down the matrix into smaller matrices until you reach the 2x2 form. The process is called expansion by minors. For example,

3 × 3 Determinant

The determinant of a 3 by 3 matrix is found by

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

A trick for this is rewriting the matrix as a 3 × 5 matrix, with copies of the first 2 columns placed after the third column:

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

and solving the *down-right diagonals* minus the *down-left diagonals*:

Down-right diagonals (add) Down-left diagonals (subtract)

$$\det(B) = (a \cdot e \cdot i) + (b \cdot f \cdot g) + (c \cdot d \cdot h) - [(c \cdot e \cdot g) + (a \cdot f \cdot h) + (b \cdot d \cdot i)]$$

Note that this is the same multiplication as above, just formatted differently.

Each term in this formula multiplies an entry by the determinant of a smaller 2 × 2 matrix. The next section extends this idea to larger matrices.

For our purposes, we simply want to first check to see if a matrix contains linearly independent rows and columns before using our Python code to solve.

2.2 Determinants in Python

NumPy computes determinants with `np.linalg.det`. Before calling `np.linalg.solve`, you can check that the system has exactly one solution. Because computers store decimals only approximately, a determinant that should be 0 may come out as something like

4×10^{-17} , so we compare against a small tolerance instead of testing for exactly zero:

```

1 import numpy as np
2
3 D = np.array([[2, 1], [1, 1]], dtype=float)
4 e = np.array([5, 3], dtype=float)
5
6 if abs(np.linalg.det(D)) > 1e-12:
7     print(np.linalg.solve(D, e))    # [2. 1.]
8 else:
9     print("Rows and columns are dependent.")

```

How does this work under the hood? Let us also write a recursive python function that finds our determinant:

There are two base cases:

- The matrix is of size 1×1
- The matrix is of size 2×2

And further sizes can be simplified into one of the base cases by *cofactor expansion*. The idea behind cofactor expansion is to break a big determinant into smaller ones until we reach cases we already know how to solve. Formally, this is written as

$$|A| = \sum_{j=1}^n (-1)^{(i+j)} a_{(ij)} M_{(ij)}$$

We do this as follows:

1. Pick the first row of the matrix.
2. For each entry in that row:
 - Remove the row and column containing that entry.
 - This creates a smaller submatrix, called a *minor*. Recall what we did above for our 3×3 determinant definition.
3. Compute the determinant of that smaller matrix.
4. Multiply the original entry (where the row and column lines originate from), the determinant of its minor, and an alternating sign $+, -, +, \dots$
5. Sum all of these results together.

This process repeats recursively: each smaller determinant is computed the same way,

until we reach one of the base cases.

If we format our matrices as a nested array, we can use python's indexing to check and reduce the matrices. Take a look at this recursive determinant program:

```

1 def recursive_determinant(matrix):
2     """
3     Compute the determinant of a square matrix recursively.
4     matrix: list of lists, forming a matrix of size n by n
5     """
6     n = len(matrix)
7
8     # base case 1x1:
9     if n == 1:
10        return matrix[0][0]
11    # base case 2x2
12    if n == 2:
13        return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]
14
15    det = 0
16
17    for col in range(n):
18        # build a minor matrix:
19        minor = []
20        # for every row in the matrix
21        for row in matrix[1:]:
22            # remove column col and store as new_row
23            new_row = row[:col]+row[col+1:]
24            # append new row to the minor matrix
25            minor.append(new_row)
26
27        sign = (-1) ** col #exponential for alternating sign
28        det += sign * matrix[0][col] * recursive_determinant(minor)
29    return det

```

We implemented our known base case, and recursively reduce our array until it fits a known base case. NumPy's `np.linalg.det()` uses a faster method called LU decomposition, but it gives the same answer as this recursion.

2.3 Inverse Matrices

Now that we have talked about determinants, we can talk about Inverse Matrices. The idea of a matrix inverse is a natural generalization of the multiplicative inverse of a real number. For example, since

$$3 \cdot \frac{1}{3} = 1,$$

the number $\frac{1}{3}$ is called the multiplicative inverse of 3.

Similarly, for matrices, we define an inverse in terms of matrix multiplication and the identity matrix. If A is an $n \times n$ matrix, its inverse (when it exists) is denoted by A^{-1} and satisfies

$$AA^{-1} = A^{-1}A = I_n,$$

where I_n is the $n \times n$ identity matrix.

Inverse Matrix

An inverse matrix is a square matrix that satisfies the following property:

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n$$

where A is the original matrix, B is the inverse matrix, and I_n is the identity matrix of size $n \times n$.

When such a matrix B exists, it is denoted by A^{-1} , and is said to be *invertible*. Note that sometimes an inverse matrix may not exist.

It is important to note that not every matrix has an inverse. In particular, only *square matrices* can be invertible, and even among square matrices, an inverse may fail to exist.

Existence of the Inverse

A square matrix A has an inverse if and only if its determinant is nonzero:

$$A^{-1} \text{ exists if and only if } \det(A) \neq 0.$$

If

$$\det(A) = 0,$$

then A is called a **singular matrix**, and no inverse exists. Equivalently, an inverse matrix does not exist when the rows or columns of A are *linearly dependent*. As discussed in the chapter on linear dependence, this occurs when one row (or column) is a scalar multiple of another, or when a row (or column) can be written as a linear combination of the others.

Linear dependence implies that the matrix does not contain enough independent information to reverse its action, making an inverse impossible.

2.3.1 Finding an Inverse Matrix

2×2 Inverse

There are several methods for finding the inverse of a matrix. For small matrices, especially 2×2 matrices, there is a direct formula. For larger matrices, a more systematic method using row operations is required.

Inverse of a 2×2 Matrix

A trick for a 2×2 Matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Note that the new matrix has a and d swapped, and b and c change signs. Finally, we multiply by $\frac{1}{ad-bc}$, the reciprocal of the determinant of the original A .

This formula is valid only when the determinant of A is nonzero. Recall that the determinant of a 2×2 matrix is

$$\det(A) = ad - bc.$$

If $ad - bc = 0$, then A is singular and has no inverse.

Worked example: 2 by 2 inverse. Find the inverse of $A = \begin{bmatrix} 3 & 1 \\ 5 & 2 \end{bmatrix}$.

Solution: First, $\det(A) = 3 \cdot 2 - 1 \cdot 5 = 1$. Swap a and d , and change the signs of b and c :

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

Dividing by $\det(A) = 1$ leaves $A^{-1} = \begin{bmatrix} 2 & -1 \\ -5 & 3 \end{bmatrix}$. To check, $AA^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Exercise 7 **2×2 Practice**

Find the inverse of the matrix of

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

Working Space

Answer on Page 75

 $n \times n$ Inverse

For matrices of size 3×3 or greater, we create an augmented matrix.

Recall that we can create an augmented matrix by writing two matrices directly next to each other.

To find the inverse of a square matrix A :

1. Form the augmented matrix $[A \mid I_n]$, where I_n is the identity matrix.
2. Use elementary row operations to transform the left side into I_n .
3. If this is possible, the right side of the augmented matrix becomes A^{-1} .

Symbolically, this process can be written as

$$[A \mid I_n] \longrightarrow [I_n \mid A^{-1}].$$

If row reduction does not result in the identity matrix on the left side, then A does not have an inverse. In this case, the matrix is singular.

Exercise 8 **A bigger inverse**

Working Space

Find the inverse of the matrix

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

Check your answer using $AA^{-1} = I_3$

Answer on Page 75

Exercise 9 Does a matrix exist?

Given the matrix:

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 2 & 4 & 8 & 16 \\ 4 & 8 & 16 & 32 \\ 8 & 16 & 32 & 64 \end{bmatrix}$$

Working Space

Does an inverse matrix exist? Explain why or why not

Answer on Page 77

The augmented matrix row-reduction method not only provides a way to compute inverses, but also offers another criterion for invertibility: a matrix is invertible if and only if it can be row-reduced to the identity matrix.

2.3.2 Relation to $A\vec{x} = \vec{b}$

An inverse matrix is also a tool for solving systems of equations, such as our fundamental equation $A\vec{x} = \vec{b}$. Let us look at an example.

Given

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$

solve for $\vec{x}$ by first finding the inverse.

First, we can interpret our givens as a systems of matrices:

$$A\vec{x} = \vec{b} \iff \begin{cases} 2x_1 + x_2 = 5 \\ x_1 + x_2 = 3 \end{cases}$$

Checking that an inverse matrix first exists, we have

$$\det(A) = 2(1) - 1(1) = 1 \neq 0$$

So we know an inverse matrix exists! To find A^{-1} :

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$$

Here is the fun part! Let us do some computational equivalences:

$$\begin{aligned} A\vec{x} &= \vec{b} \\ A^{-1}(A\vec{x}) &= A^{-1}\vec{b} \\ I\vec{x} &= A^{-1}\vec{b} \\ \vec{x} &= A^{-1}\vec{b} \end{aligned} \tag{2.1}$$

Equation (2.1) shows that, by only multiplying by the matrix equivalent of 1, the identity matrix I , we can state that $\vec{x} = A^{-1}\vec{b}$. Since we have both A^{-1} and $\vec{b}$, we can find $\vec{x}$:

$$\vec{x} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Geometrically, this did a very simple operation. We used the fact that the matrix A transforms $\vec{x}$ into $\vec{b}$. Then we noted that A^{-1} undoes that transformation, recovering our original $\vec{x}$ by finding $A^{-1}\vec{b}$. This may be hard to visualize, so do not worry if it is hard to grasp immediately. We will review it again in a following chapter.

2.3.3 Matrix Inverse using NumPy

We can use our Python library NumPy to find inverses much more easily using the `np.linalg.inv` function to find the inverse of a given array.

```
1 import numpy as np
2
```

```

3 A = np.array([[2, 1],
4               [1, 1]])
5
6 A_inv = np.linalg.inv(A)
7 print(A_inv)

```

This outputs

```

1 [[ 1. -1.]
2  [-1.  2.]]

```

If given a matrix that is *linearly dependent*, NumPy will raise an exception:

```

1 B = np.array([[2, 4],
2               [1, 2]])
3 np.linalg.inv(B)

```

The last line reports the error, which tells us that the inverse does not exist.

To improve our code, we check for the determinant of the matrix beforehand. In this code, we use a tolerance because computers store numbers close to zero only approximately.

```

1 import numpy as np
2
3 A = np.array([[2, 1],
4               [4, 2]], dtype=float)
5
6 # BAD PRACTICE:
7 # A_inv = np.linalg.inv(A)
8 # print(A_inv)
9
10 detA = np.linalg.det(A)
11
12 tolerance = 1e-12
13
14 if abs(detA) < tolerance:
15     print("Matrix is singular or nearly singular. No reliable inverse.")
16 else:
17     A_inv = np.linalg.inv(A)
18     print("Determinant:", detA) # you may see a large floating point number
19     print("Inverse:\n", A_inv)

```

Output:

```

1 Matrix is singular or nearly singular. No reliable inverse.

```

2.4 Summary

In two dimensions, the determinant of a 2×2 matrix represents the signed area of the parallelogram formed by the images of the standard basis vectors. In higher dimensions, the determinant represents signed volume.

Further, the absolute value $|\det(A)|$ is the factor by which a matrix scales areas or volumes. The sign of $\det(A)$ tells us whether the matrix flips orientation.

The inverse of a matrix *reverses the effect of the original matrix*. If A is invertible, then multiplying by A^{-1} restores any vector to its original position:

$$AA^{-1} = A^{-1}A = I.$$

Thus, invertibility, nonzero determinant, linear independence of rows and columns, and reversibility of action are all different ways of describing the same underlying property.

In the next chapter, we will treat matrices as *functions* that transform space: stretching, rotating, and flipping it. Determinants will measure how much area changes, and inverses will undo the transformation.

2.5 Where to Learn More

The Wolfram Demonstrations website has an interactive demo where you can enter 2D and 3D matrices and watch the area or volume change: <https://demonstrations.wolfram.com/DeterminantsSeenGeometrically/>

For more on expansion by minors, see <https://mathworld.wolfram.com/DeterminantExpansionbyMinors.html>

Matrices as Transformations

When you pinch to zoom on a photo, rotate your phone, or watch a video-game character turn, every point on the screen is being fed through a 2×2 matrix. This chapter shows how.

Recall that a *function*, informally, is a rule that takes an input and produces an output.

For example, we know the common function $f(x) = x^3$ takes the input and cubes it, such that the result is $x \times x \times x$. Inputting $x = 3$ outputs $f(3) = 27$.

In Linear Algebra, we can think of matrices as a type of function. Recall our systems of matrices equation,

$$A\vec{x} = \vec{b}$$

We can rearrange this to look closer to function notation:

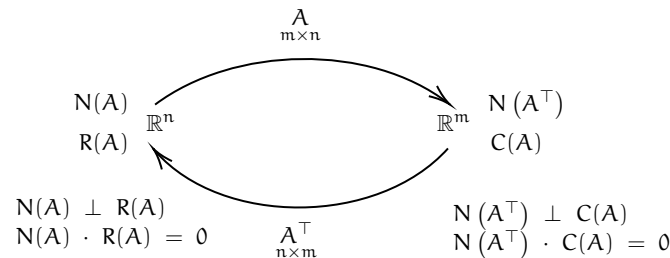
$$\vec{b} = A\vec{x}$$

We can change our input vector $\vec{x}$, which directly affects the output variable $\vec{b}$.

Recall our subspace diagram; the matrix A , which is size $m \times n$, takes the input vectors $\vec{x} \in \mathbb{R}^n$, and transforms them to the output $\vec{b} \in \mathbb{R}^m$.

The transformation T maps $\mathbb{R}^n$ to $\mathbb{R}^m$, written as $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$

- $\mathbb{R}^n$ is called the **domain**
- $\mathbb{R}^m$ is called the **codomain**



Let us call the transformation T , such that

$$T : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

We can rewrite our transformation with matrix A

$$T(\vec{x}) = A\vec{x}$$

A matrix transformation is completely determined by where it sends the *standard basis vectors*. Each column of A is the image of a basis vector under the transformation, and every output vector is a linear combination of these columns.

Consequently, the column space of A represents the full set of possible outputs of the transformation.

3.0.1 The Identity Transformation

For this chapter, we will restrict our transformations to $\mathbb{R}^2 \rightarrow \mathbb{R}^2$, such that our matrix is 2×2 . This will help calculations be simpler and also will allow us to make easy to follow transformation diagrams.

Let us take the simplest transformation; the *identity transformation*. Take the matrix:

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

Every vector in $\mathbb{R}^n$ **stays where it is**.

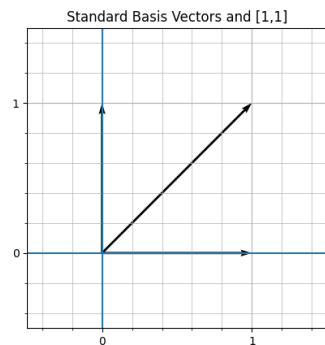
- e_1 stays e_1
- e_2 stays e_2

Every vector stays where it originally started in this transformation. In a way, no transformation is truly applied.

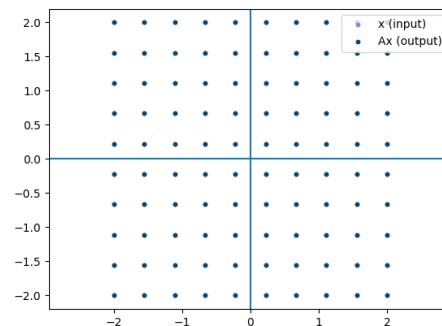
Under the identity transformation, nothing moves. We still get something very important out of this: the vector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ can be written as a combination of the basis vectors:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1\vec{e}_1 + 1\vec{e}_2$$

When a linear transformation is applied, the way a vector is built from the basis vectors does not change, but *the directions of the basis vectors* can change.



(a) Identity vectors before transformation



(b) Identity vectors after transformation (no change)

Figure 3.1: An identity transformation acting on basis vectors.

A linear transformation acts on $\vec{x}$ by acting on each basis vector individually:

$$A\vec{x} = x_1 A\vec{e}_1 + x_2 A\vec{e}_2$$

such that any coefficient x_1 and x_2 are unchanged, while $\vec{e}_1$ and $\vec{e}_2$ are transformed.

Our chapter will also take a look at transforming a simple house. Here is the house before any transformations (or, with the identity transformation applied):

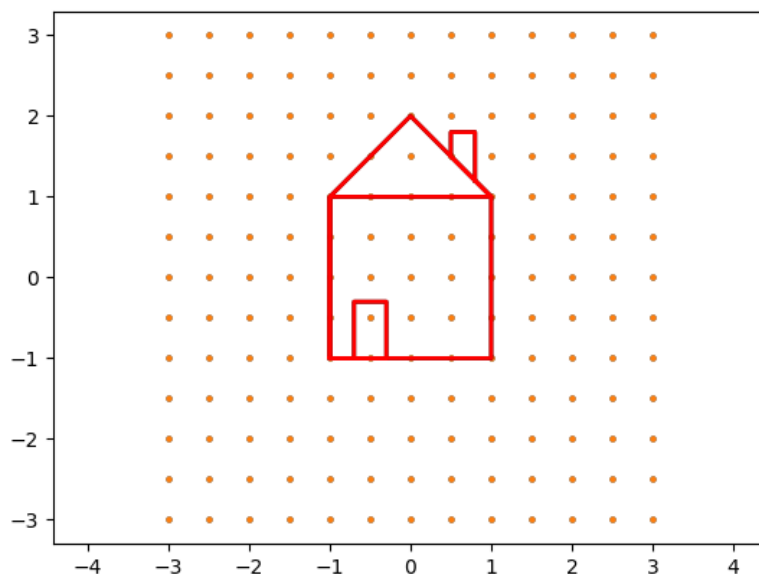


Figure 3.2: A simple house before any transformations.

Linear transformations preserve linear combinations while *altering the directions that define the coordinate system*.

In the next example, we will apply a non-identity matrix and observe how transforming the basis vectors reshapes the entire space. Each figure will show a grid of points in $\mathbb{R}^2$ alongside the resulting transformed points after applying the matrix transformation. We encourage you to play with the python script `transforms.py` to see how different matrices affect the transformation. Also, we will limit our view to a square region from $(-2, -2)$ to $(2, 2)$ and only use 2×2 matrices for simplicity.

3.1 Scaling and Shearing

Now, let us try scaling up and down the basis vectors. Consider the matrix:

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

This matrix will scale both the e_1 and e_2 vectors by a factor of 2, effectively doubling their lengths.

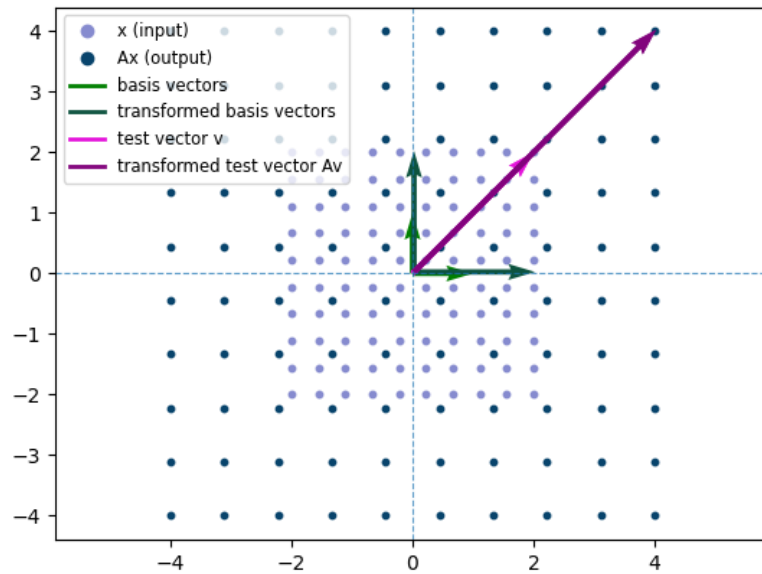


Figure 3.3: The effect of the scaling transformation $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ on a grid of points.

What happens to our house? Well, each part of the house is transformed linearly by 2, a linear *scaling*.

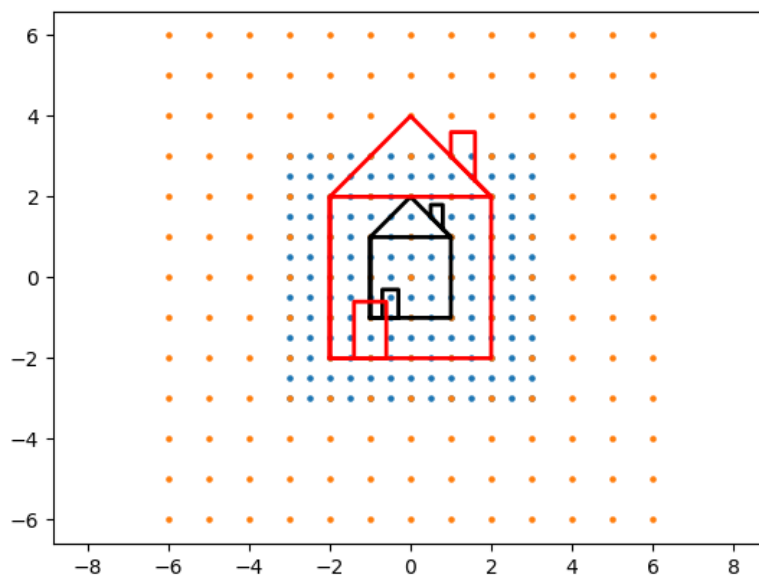


Figure 3.4: The effect of the scaling transformation $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ on a simple house.

This tells us that the transformation A is a scaling transformation, which stretches the space by a factor of 2 in both directions. What do you think happens if we scale by a factor of $\frac{1}{2}$ instead? Try it out!

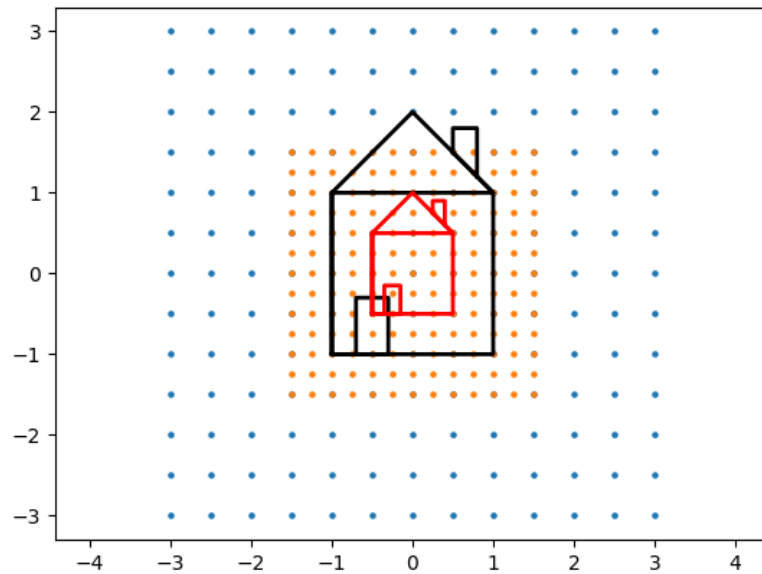


Figure 3.5: The effect of the scaling transformation $A = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ on a simple house.

This, as expected, shrinks the house by a factor of $\frac{1}{2}$ in both directions.

Now, let us try scaling up *only one* of the basis vectors. Consider the matrix:

$$A_{\text{vertical scaling}} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

This matrix will scale the e_1 vector by a factor of 1 (no change) and the e_2 vector by a factor of 2 (doubling its length).

Applying this transformation to the basis vectors produces this result:

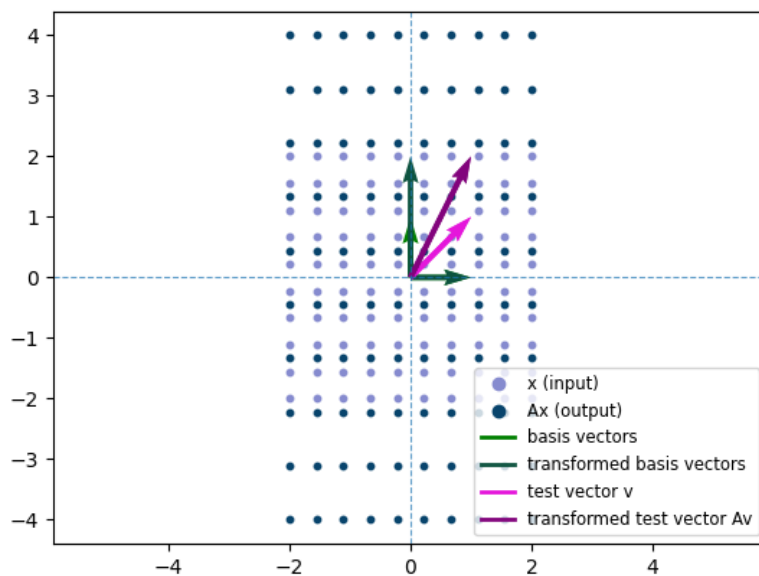


Figure 3.6: The effect of the scaling transformation $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ on the standard basis vectors.

A shear slides one side of a shape while the other side stays fixed. The matrix $\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$ is a shear: it keeps $\vec{e}_1$ fixed and sends $\vec{e}_2$ to $(2, 1)$. Applying this shear to our house gives:

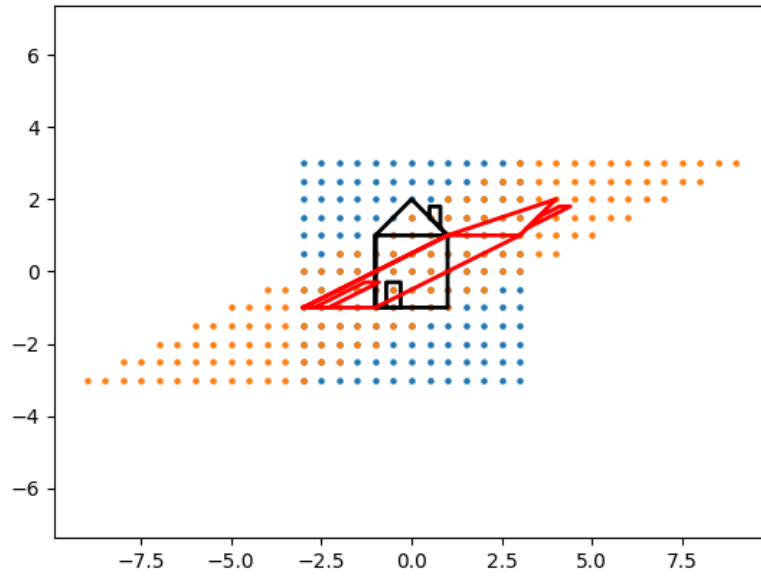


Figure 3.7: The effect of the shearing transformation $A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$ on a simple house.

3.2 Reflections

How would we do reflections across a given axis? Let us think about what happens to each point. Reflecting individual points is the same as reflecting the basis vectors, since every point is a linear combination of the basis vectors.

3.2.1 Reflection across the x-axis

For a reflection across the x-axis, the y-values of each point change sign, while the x-values stay the same.

$$(x, y) \mapsto (x, -y)$$

This means that e_1 stays e_1 , while e_2 becomes $-e_2$. This allows us to rewrite a 2×2 matrix for this transformation:

$$A_{x\text{-axis}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Below is a pink test vector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and its transformation in purple, which is $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

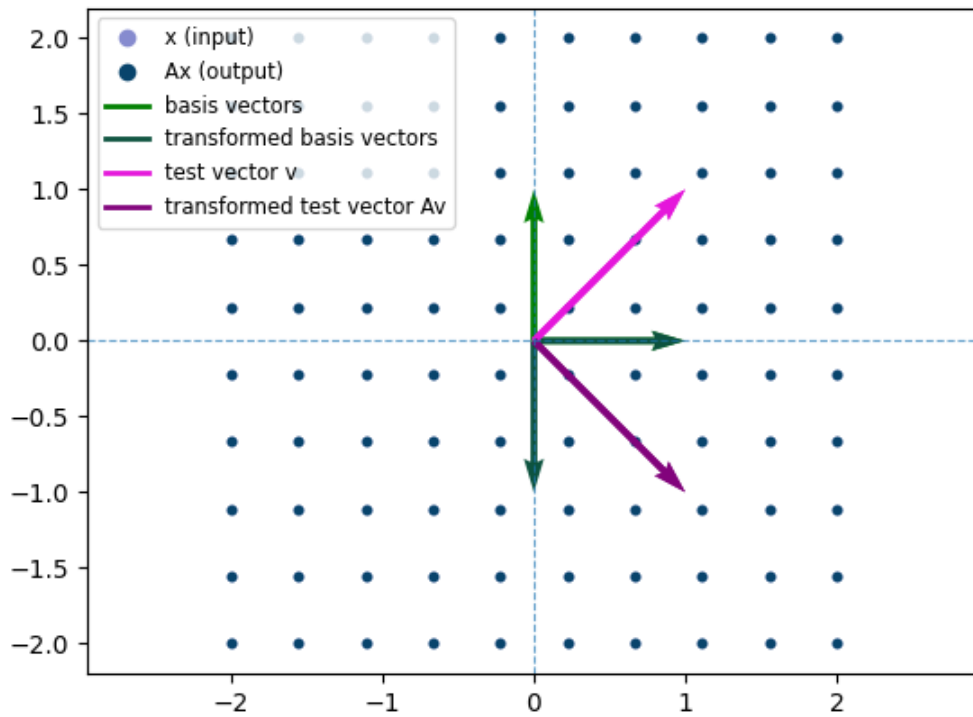


Figure 3.8: The effect of reflecting across the x-axis. Added is the vector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and its transformation in purple.

Also, we can reflect our house across the x-axis. We will see a difference in where the chimney, roof, and door are located.

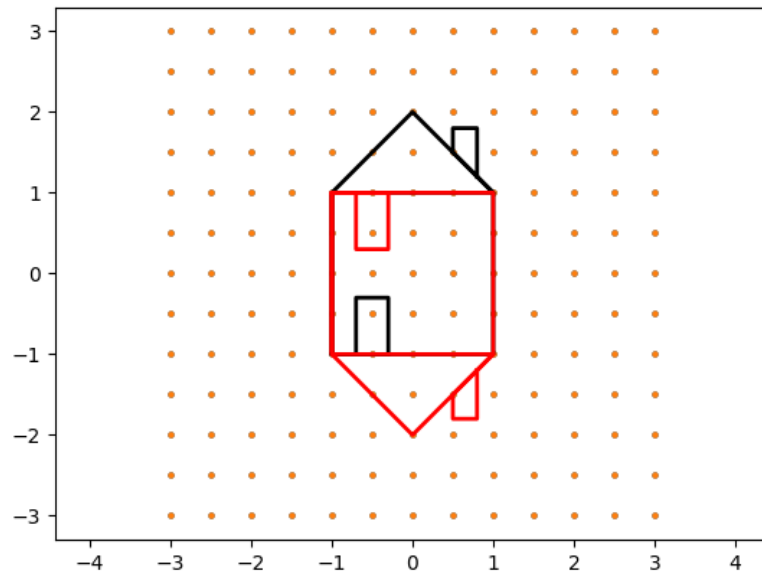


Figure 3.9: The effect of reflecting a house across the x -axis. Red house is the transformed house, while the black house is the original.

3.2.2 Reflection across the y -axis

What if we want to reflect across the y -axis instead? In this case, the x -values of each point change sign, while the y -values stay the same. As a function of points, this is

$$(x, y) \mapsto (-x, y)$$

Our basis vector e_1 becomes $-e_1$, while e_2 stays e_2 . This allows us to rewrite a matrix for this transformation:

$$A_{y\text{-axis}} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

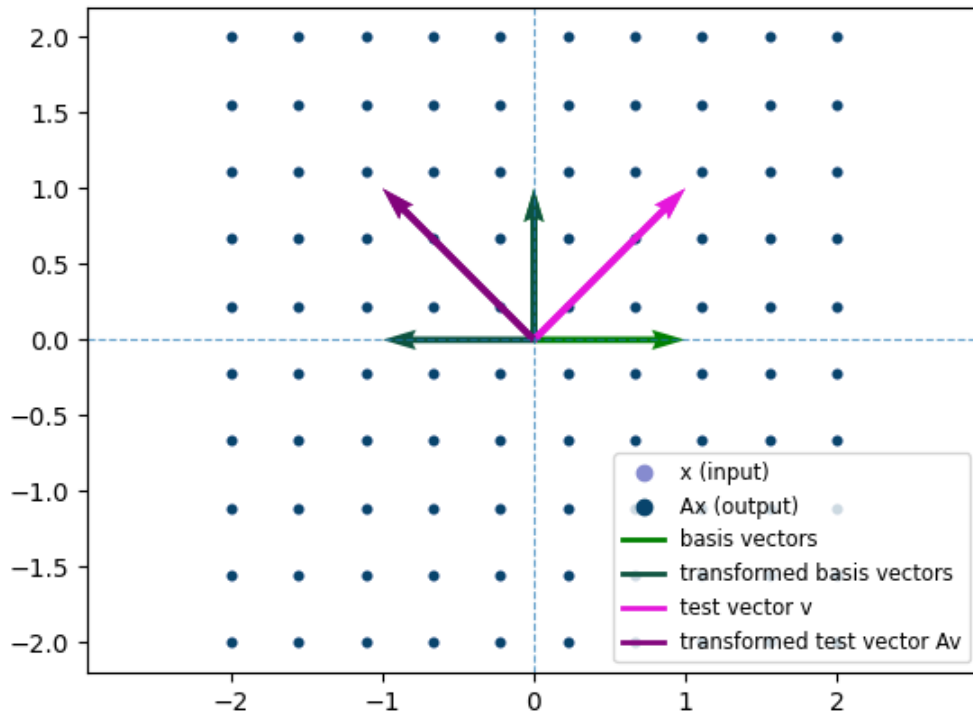


Figure 3.10: The effect of reflecting across the y -axis. Added is the vector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and its transformation in purple.

Notice that it is similar to the effect of a selfie, where the left and right sides are flipped. This may be more noticeable in the house transformation, where the door and chimney are on the opposite sides of the house.

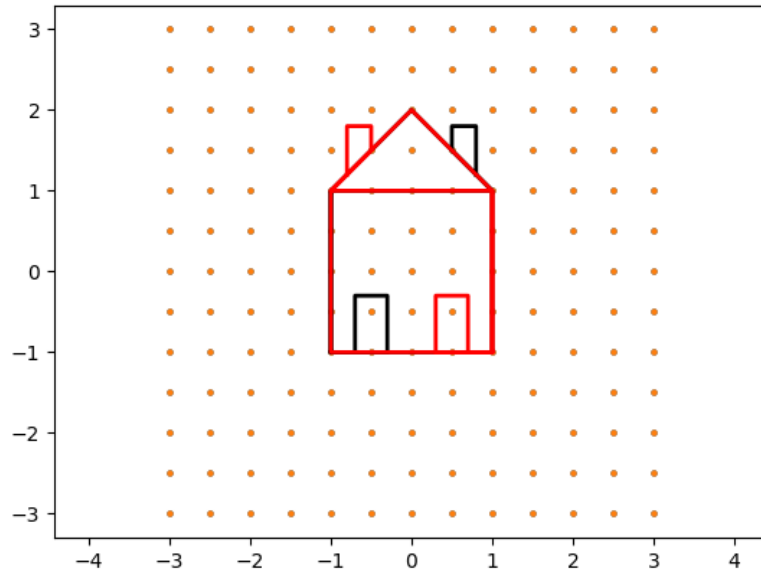


Figure 3.11: The effect of reflecting a house across the y -axis.

Note that the red house is still upright, but the left and right sides are flipped. This makes sense, since the y -values of the points are unchanged, while the x -values flip signs.

3.2.3 Reflection across the lines $y = x$ and $y = -x$

Two other useful lines are $y = x$ and $y = -x$. We can also reflect across these lines.

The line $y = x$ is the line where the x and y values are equal. When we reflect across this line, the x and y values swap places. As a function of points, this is

$$(x, y) \mapsto (y, x)$$

Since each point swaps places, the basis vectors also swap places. e_1 becomes e_2 , while e_2 becomes e_1 . This allows us to rewrite a matrix for this transformation:

$$A_{y=x} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Let us look at our house picture, reflected across the line $y = x$. Notice that the house is flipped across the line, such that the roof is now on the bottom and the door is on top.

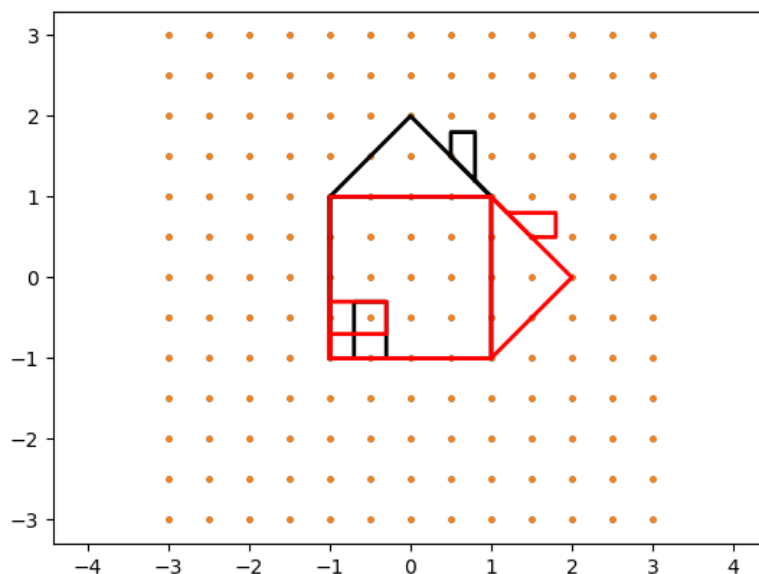


Figure 3.12: The effect of reflecting a house across the line $y = x$.

The line $y = -x$ is the line where the x and y values are equal but with opposite signs. When we reflect across this line, the x and y values swap places and flip signs. As a function of points, this is

$$(x, y) \mapsto (-y, -x)$$

Not only do the x and y values swap places, but they also flip signs. This means that e_1 becomes $-e_2$, while e_2 becomes $-e_1$. This allows us to rewrite a matrix for this transformation, where

$$A_{y=-x} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

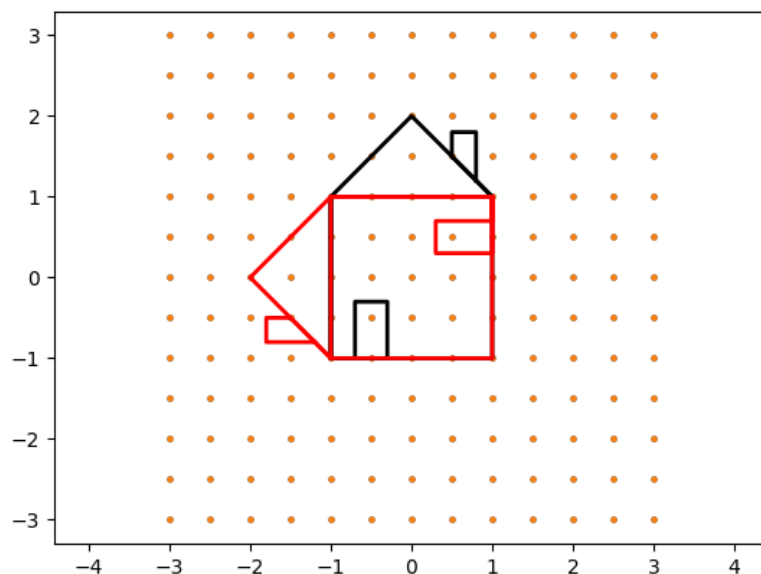


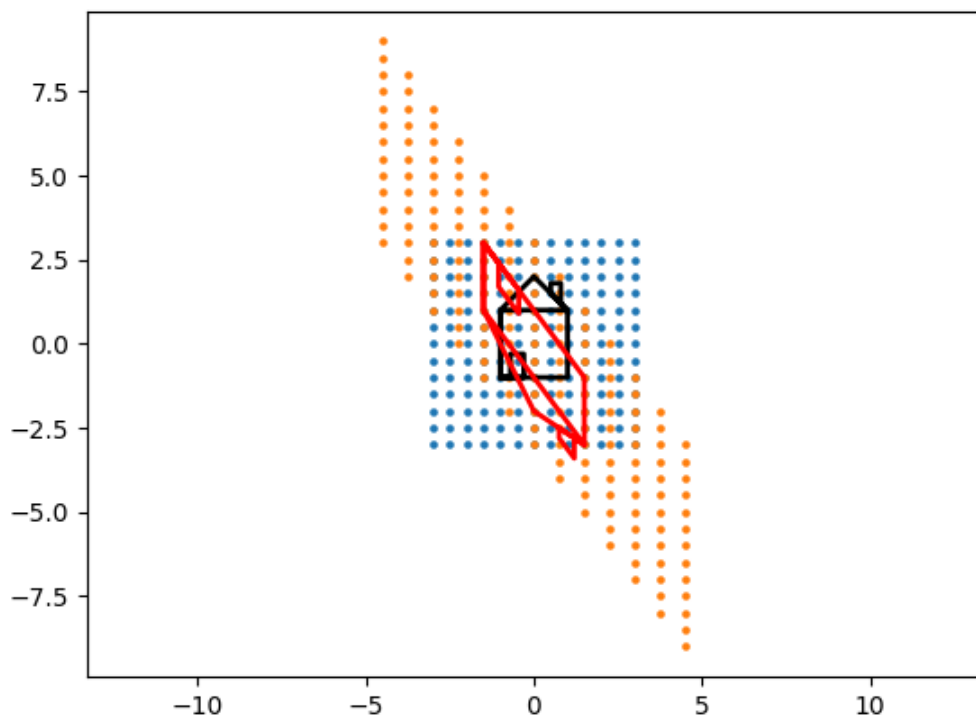
Figure 3.13: The effect of reflecting a house across the line $y = -x$.

Exercise 10 Guess the transformation matrix

Given the following transformation of the house, find the matrix that represents this transformation.

Hint: the blue grid runs from -3 to 3 in each direction, with 13 points in each row and column (spaced 0.5 apart). Track where the corner $(3, 3)$ ends up.

Working Space



Answer on Page 77

3.3 Rotations

The four reflections above each flip the house across a line.

What if we want to rotate our house, like turning the hands of a clock? We can also do this with a matrix transformation. It may look similar to the reflections, but it is not the same. Let us track our basis vectors to see where they end up:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

From this, we can write the matrix for a 90° rotation clockwise:

$$A_{90^\circ}^{\text{cw}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

What about a 90° rotation counterclockwise? We can track our basis vectors again:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

And so the matrix for a 90° rotation counterclockwise is:

$$A_{90^\circ}^{\text{ccw}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Worked example. Apply the counterclockwise matrix to the point $(2, 1)$:

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

The point moves to $(-1, 2)$, a quarter turn counterclockwise.

However, we have learned trigonometric functions that result in these values. We know that at 90° , $\cos(90^\circ) = 0$ and $\sin(90^\circ) = 1$. So we could say:

$$A_{90^\circ}^{\text{cw}} = \begin{bmatrix} \cos(90^\circ) & \sin(90^\circ) \\ -\sin(90^\circ) & \cos(90^\circ) \end{bmatrix}$$

Note that this aligns with our *polar* coordinate system where cosine is the x-value and sine is the y-value.

For a counterclockwise rotation of 90° , we have:

$$A_{90^\circ}^{\text{ccw}} = \begin{bmatrix} \cos(90^\circ) & -\sin(90^\circ) \\ \sin(90^\circ) & \cos(90^\circ) \end{bmatrix}$$

But, we can generalize this to any angle θ instead of just 90° . For a clockwise rotation by θ , we have:

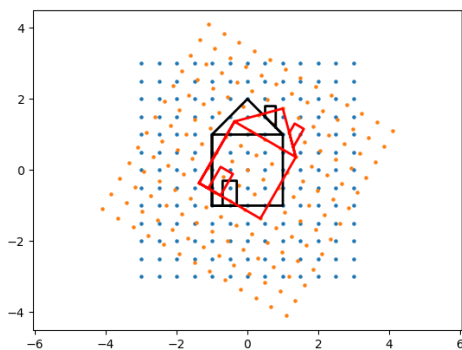
$$A_{\theta}^{\text{cw}} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

And for a counterclockwise rotation by θ , we have:

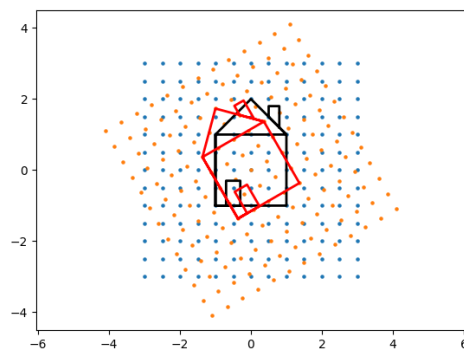
$$A_{\theta}^{\text{ccw}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Note how similar they are! The only difference is the sign of the sine values. Sine is an odd function, which means that $\sin(-\theta) = -\sin(\theta)$, so the sign of the sine values determines the direction of rotation.

Let us see what happens when we apply a 30° clockwise and counterclockwise rotation to our house. We can calculate the matrix for this transformation using the cosine and sine values for 30° :



(a) 30 degree clockwise rotation



(b) 30 degree counterclockwise rotation

Figure 3.14: The effect of 30 degree rotations on our house.

3.4 What if I want to undo my transform?

We have talked about how to apply many transformations, but what if we want to undo a transformation? For example, if we rotate a house 30° clockwise (3.14a), how do we get it back to its original position?

This question brings us back to the application of those *inverse matrices* that we learned about in the previous chapters. If we have a transformation represented by a matrix A , then the inverse transformation is represented by the inverse matrix A^{-1} . Applying A^{-1} to the transformed house will return it to its original position.

Additionally, given a transformed basis vector set $\vec{e}_1'$ and $\vec{e}_2'$, we can apply the inverse transformation to find the original basis vectors. This is because the inverse transformation will reverse the effects of the original transformation, allowing us to recover the original basis vectors from their transformed counterparts.

If you recall, for our 2×2 matrices, the inverse is simple! For a matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, the inverse is

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Any larger matrices can get convoluted, so we use gaussian elimination to find the inverse or our numpy function `np.linalg.inv()`.

The inverse transformation of our 30° clockwise rotation is going to be

$$\begin{aligned} A_{30^\circ \text{cw}}^{-1} &= \frac{1}{\cos(30^\circ)\cos(30^\circ) - (-\sin(30^\circ))\sin(30^\circ)} \begin{bmatrix} \cos(30^\circ) & -\sin(30^\circ) \\ \sin(30^\circ) & \cos(30^\circ) \end{bmatrix} \\ &= \frac{1}{\cos^2(30^\circ) + \sin^2(30^\circ)} \begin{bmatrix} \cos(30^\circ) & -\sin(30^\circ) \\ \sin(30^\circ) & \cos(30^\circ) \end{bmatrix} \\ &= \frac{1}{1} \begin{bmatrix} \cos(30^\circ) & -\sin(30^\circ) \\ \sin(30^\circ) & \cos(30^\circ) \end{bmatrix} \\ &= \begin{bmatrix} \cos(30^\circ) & -\sin(30^\circ) \\ \sin(30^\circ) & \cos(30^\circ) \end{bmatrix} \end{aligned}$$

Notice that this is the same as the matrix for a 30° counterclockwise rotation! This makes sense, since a 30° counterclockwise rotation would undo a 30° clockwise rotation.

Note that if the determinant is 0, then the associated linear transformation is not invertible. In this case, the transformation collapses space into a lower dimension (for example, a plane into a line or a line into a point), causing information to be lost. As a result, the transformation cannot be reversed, and therefore no inverse matrix exists.

Common examples of this are:

$$\begin{aligned} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \text{ (projection matrix (covered in a future chapter))} \\ \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix} & \text{ (eliminates one direction)} \\ \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} & \text{ (duplicate / linearly dependent rows)} \end{aligned}$$

Exercise 11 Undoing a Shear

Working Space

A shear, by definition, leaves one axis fixed and slides points parallel to it. For example, an x-shear is given by the following relationship:

$$(x, y) \mapsto (x + ky, y)$$

1. Find the original matrix for an x-shear

with a shear factor of k .

2. Find the inverse matrix for this transformation.

Answer on Page 77

Exercise 12 Build a Matrix

Write the matrix that first reflects across the x -axis and then doubles every length. Where does the point $(1, 2)$ go? What is the determinant, and what does it tell you?

Working Space

Answer on Page 78

3.5 Summary and looking ahead

In this chapter, we have seen how matrices can be used to represent transformations in $\mathbb{R}^2$. We have explored various types of transformations, including scaling, shearing, reflections, and rotations. We have also discussed how to find the inverse of a transformation, which allows us to undo the effects of a transformation.

We want to look at these transformation as effects on directions. Specifically, are there any directions that this transformation does not mix with others?

- Reflections have the property that they flip one direction while keeping the other direction unchanged. For example, a reflection across the x -axis flips the y -direction while keeping the x -direction unchanged. In this, some directions stay the same, while others are flipped. Certain directions are preserved, while others are not.
- Rotations, on the other hand, mix all directions together. No direction stays on its own line under a rotation, except in the special cases of 0° (nothing moves) and 180° (every vector is reversed).
- Scalings stretch or shrink certain directions without turning them. The matrix $\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

leaves the x-direction alone and doubles the y-direction.

For some transformations, there exist special directions along which vectors are simply scaled. In studying linear transformations, it is natural to ask whether there exist directions that remain unchanged, except for scaling of its length. Before reading the next chapter, try to find a nonzero vector whose direction is *unchanged* by each transformation. Not every transformation will have such a vector, but some will.

Eigenvectors and Eigenvalues

When a bridge sways in the wind, a guitar string vibrates, or Google ranks web pages, the math underneath is the same: eigenvectors. They are the directions a transformation leaves alone.

Eigen is German for “own” or “characteristic.” Named by the German mathematician David Hilbert, an eigenvector mathematically describes a characteristic feature of an object that remains unchanged after transformation. You can think of an eigenvector as a direction that does not change under a transformation. Each eigenvector has an eigenvalue, which tells how much the vector is scaled. Eigenvalues can be negative or positive. A negative value indicates the direction of the eigenvector is reversed.

Eigenvalues and eigenvectors are a way to break down matrices, which can simplify many calculations and enable us to understand various properties of the matrix. They are widely used in physics and engineering for stability analysis, vibration analysis, and many other applications.

Let us look at a visual example.

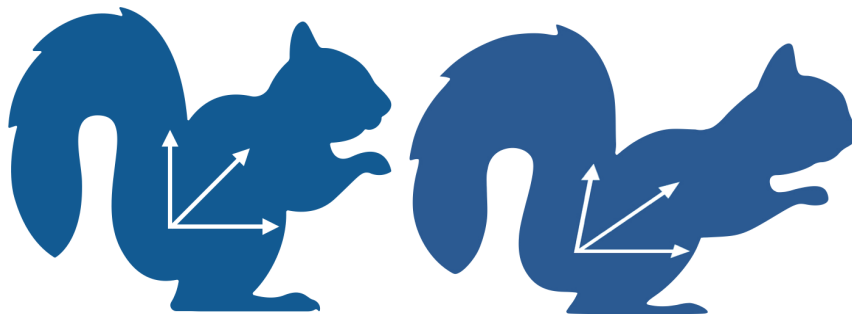


Figure 4.1: Standardized and skewed squirrel image.

You can see that the image on the right is a skewed version of the image on the left. Look closely at the vectors and you will notice that one of the vectors is pointing in the same direction in both images, while the direction of the other two vectors has changed. The eigenvector is the one at the bottom that points to 0 degrees (which you can think of as due east) in both images. So, the characteristic attribute of both images is their horizontal direction.

Think about a rubber sheet with a grid drawn on it. When you pull the sheet, most lines

tilt. A line that you pull straight along stays on its own line, and only its length changes. That line is an eigenvector.

When you overlay the vectors from one image over the other, you will notice that the horizontal vector, while the same direction in both images, is a bit longer in the skewed version. The scale of the stretch is described by an eigenvalue.

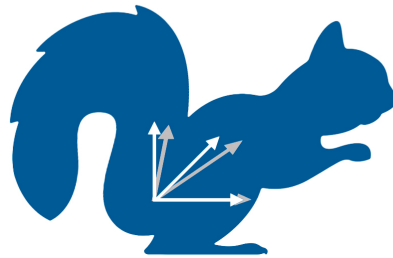


Figure 4.2: Standardized and skewed squirrel image overlaid on each other.

4.1 Definition

Eigenvector and Eigen Equation

Given a square matrix A , a non-zero vector v is an eigenvector of A if multiplying A by v results in a scalar multiple of v . In other words, the *eigenequation* is:

$$Av = \lambda v \quad (4.1)$$

where λ is a scalar known as the eigenvalue corresponding to the eigenvector v .

An eigenvector is a nonzero vector that does not change direction when a linear transformation is applied.

It may stretch, shrink, or flip direction, but it *stays on the same line*.

4.2 Finding Eigenvalues and Eigenvectors

You find the eigenvalues of a matrix A by solving the characteristic equation:

$$\det(A - \lambda I) = 0 \quad (4.2)$$

where $\det(\cdot)$ denotes the determinant, I is the *identity matrix* of the same size as A , and λ is a scalar. In other words, we subtract λ from each diagonal entry of A , and solve for the determinant.

$$A - \lambda I = \begin{bmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{bmatrix}$$

Once you find the eigenvalues, you can find the corresponding eigenvectors by substituting each eigenvalue into the equation $Av = \lambda v$, and solving for v . v is a vector which only gets *scaled*, with no change in direction.

For a non-zero v to satisfy this equation,

- $\det(A - \lambda I) = 0$ must hold
- $A - \lambda I$ cannot have an inverse
- the transformation squashes space onto a set of zero volume.

4.3 Example

For a 2×2 matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the characteristic equation is:

$$(a - \lambda)(d - \lambda) - bc = 0 \quad (4.3)$$

Solving this equation gives the eigenvalues. Substituting each eigenvalue back into the equation $Av = \lambda v$ gives the corresponding eigenvectors.

Let matrix $A =$

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

The characteristic equation is:

$$\begin{aligned} |A - \lambda I| &= 0 \\ \begin{vmatrix} 5 - \lambda & 4 \\ 1 & 2 - \lambda \end{vmatrix} &= 0 \\ (5 - \lambda)(2 - \lambda) - (4)(1) &= 0 \\ 10 - 5\lambda - 2\lambda + \lambda^2 - 4 &= 0 \end{aligned}$$

$$\lambda^2 - 7\lambda + 6 = 0$$

$$(\lambda - 6)(\lambda - 1) = 0$$

$$\lambda = 6, \lambda = 1$$

Now that you have the eigenvalues, substitute each one into $(A - \lambda I)\vec{v} = \vec{0}$ and solve for $\vec{v}$.

For $\lambda = 1$:

$$(A - \lambda I)v = \vec{0}$$

$$\begin{bmatrix} 5-1 & 4 \\ 1 & 2-1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 4 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Next, use elementary row transformation by multiplying row 2 by 4, then subtracting row 1.

$$\begin{bmatrix} 4 & 4 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Now you can expand as an equation:

$$4x + 4y = 0$$

Assume $y = w$

$$4x = -4w$$

$$x = -w$$

The solution is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -w \\ w \end{bmatrix} = w \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

So the eigenvector is:

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Now we need to substitute the other eigenvalue, 6, into the equation and follow the same procedure for finding the eigenvector.

$$\begin{bmatrix} 5-6 & 4 \\ 1 & 2-6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 4 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Next, use elementary row transformation by adding row 1 to row 2.

$$\begin{bmatrix} -1 & 4 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Expand as an equation:

$$-x + 4y = 0$$

Let $y = w$:

$$-x + 4w = 0$$

$$x = 4w$$

The solution is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4w \\ w \end{bmatrix} = w \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

So the eigenvector is:

$$\begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

In conclusion, the eigenvectors of the given 2×2 matrix are:

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

4.4 Eigenbasis and Eigenspace

Let A be a square matrix and let λ be an eigenvalue of A .

Eigenspace

The **eigenspace** corresponding to λ is the set of all vectors v such that

$$Av = \lambda v.$$

Equivalently, the eigenspace of λ is the solution set of

$$(A - \lambda I)v = 0.$$

Note that there will likely be multiple lambdas for a given matrix, and therefore multiple eigenspaces for each matrix.

The eigenspace is a subspace consisting of all eigenvectors associated with λ , together including the zero vector, $\vec{0}$.

Eigenbasis

An *eigenbasis* for a matrix A is a basis of the vector space consisting entirely of eigenvectors of A .

$$\{v_1, v_2, \dots, v_n\}$$

is an eigenbasis for A if:

$$Av_i = \lambda_i v_i \quad \text{for each } i = 1, 2, \dots, n$$

Exercise 13 **A 2×2 matrix**

Working Space

Find the eigenbasis of the matrix

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

Answer on Page 78

4.5 Eigenvalues and Eigenvectors in Python

NumPy's `np.linalg.eig` finds eigenvalues and eigenvectors for you. Let us check the first example in this chapter. Create `eigen.py`:

```
1 import numpy as np
2
3 A = np.array([[5, 4],
4              [1, 2]])
5 values, vectors = np.linalg.eig(A)
6
7 print("Eigenvalues:", values)
8 print("Eigenvectors (one per column):")
9 print(vectors)
10
11 # Check the first pair: A v should equal lambda v
12 v = vectors[:, 0]
13 print(A @ v, values[0] * v)
```

You should see:

```
1 Eigenvalues: [6. 1.]
2 Eigenvectors (one per column):
3 [[ 0.9701425 -0.70710678]
4  [ 0.24253563  0.70710678]]
5 [5.820855  1.45521375] [5.820855  1.45521375]
```

(Some versions of NumPy print these as complex numbers, like `6.+0.j`; the `+0.j` just means the imaginary part is zero.)

NumPy scales each eigenvector to length 1. The first column, `[0.970, 0.243]`, is $\frac{1}{\sqrt{17}}[4, 1]$: the same direction as the eigenvector `[4, 1]` we found by hand. The second is $\frac{1}{\sqrt{2}}[-1, 1]$. Remember that any nonzero multiple of an eigenvector is also an eigenvector.

4.6 Connection to transformations

In the previous chapter, we studied matrix transformations such as scaling, reflection, shear, and rotation. Each of these transformations changes vectors in space by stretching, shrinking, flipping, or rotating them. Most vectors change both their length and/or their direction under such transformations.

Eigenvectors arise naturally when a transformation leaves at least one direction unchanged. Different types of transformations exhibit different eigenvector behavior:

- **Uniform scaling** (kI) preserves all directions, so every nonzero vector is an eigenvector. A scaling that stretches only one axis keeps the two axis directions.
- **Reflections** preserve the direction of vectors along the axis of reflection and reverse vectors perpendicular to it. Vectors lying on the axis of reflection are unchanged by the transformation and therefore are eigenvectors with eigenvalue 1. Vectors perpendicular to the axis of reflection are flipped to point in the opposite direction, making them eigenvectors with eigenvalue -1 .
- **Shear transformations** preserve exactly one direction, even though most vectors change direction.
- **Rotations in two dimensions** generally preserve no directions, and therefore have no real eigenvectors. A 90° rotation has none. The exceptions are $\theta = 0$, where every vector is an eigenvector with eigenvalue 1, and $\theta = \pi$, where every vector has eigenvalue -1 .

These observations explain why some matrices have many eigenvectors, some have only a few, and some have none at all. Eigenvectors exist precisely when a transformation has invariant directions.

Understanding eigenvectors as preserved directions allows us to connect algebraic computations to geometric behavior. When a matrix has enough independent eigenvectors, it becomes possible to describe the transformation entirely in terms of simple scaling along these special directions.

4.7 Summary

In summary, let us take a *transformation*:

Something that stretches, squishes, flips, or rotates space. This transformation could be represented by a matrix.

Most vectors in space will get changed in both direction and length when you apply the transformation.

But some special vectors do not change direction at all — they *only* get stretched or shrunk (and maybe flipped).

These special “unchanging-direction” vectors are called eigenvectors. The amount they get stretched (or shrunk) is their eigenvalue.

Exercise 14 An upper triangular matrix

Find the eigenvalues and an eigenvector

Working Space

for each eigenvalue of

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

Check one of your eigenvectors by computing A times that vector.

Answer on Page 79

4.8 Where to Learn More

Watch this video from Khan Academy, *Introduction to Eigenvectors*: <https://rb.gy/mse7i>

Projections

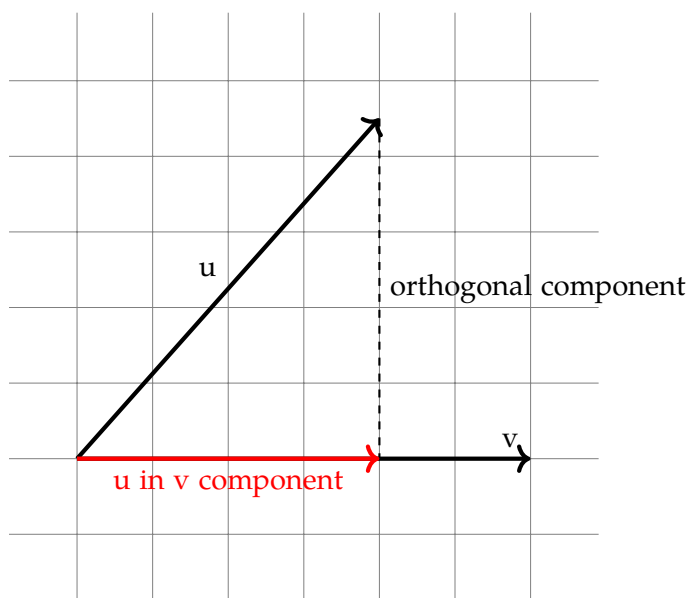
At noon, with the sun straight overhead, your shadow is a small blob at your feet. Hold a meter stick at an angle, and its shadow on the ground shows how much of the stick points sideways. That shadow is a *projection*: the part of one thing that lies along another.

Projections are used in many fields, such as science, math, engineering, and finance. Here are a few examples:

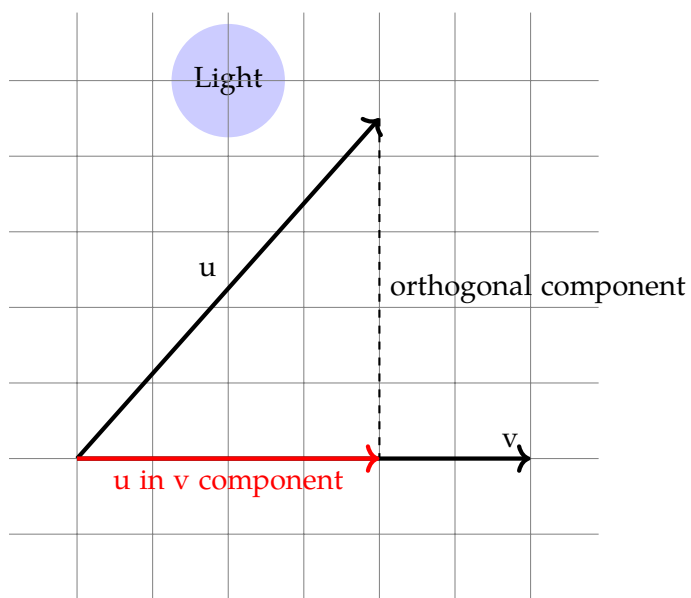
- Investors evaluate risk and return of a portfolio by projecting an asset's return onto a reference portfolio.
- Astronomers analyze the motion of stellar objects by projecting the object's true motion onto the plane of the sky.
- Robotics engineers use projections to prevent robots from running into obstacles by projecting the robot's position onto the optimal path.

Mathematically, a projection describes the relationship of one vector to another in terms of direction and orthogonality. Given two vectors, $\mathbf{u}$ and $\mathbf{v}$, the projection of $\mathbf{u}$ onto $\mathbf{v}$ separates $\mathbf{u}$ into two components. The first component signifies how much $\mathbf{u}$ lies in the direction of $\mathbf{v}$. The second signifies the component of $\mathbf{u}$ that is orthogonal (perpendicular) to $\mathbf{v}$.

The figure 5.1 depicts a projection. The perpendicular line dropped from the end of $\mathbf{u}$ is the *orthogonal* component. The portion of $\mathbf{u}$ that lies in the direction of $\mathbf{v}$ is the blue segment.

Figure 5.1: The projection of u onto v

You can also think of a projection as the shadow cast by one vector onto the other by an overhead light. See Figure 5.2

Figure 5.2: Projection of u onto v with a light included to simulate a shadow.

The projection always lies on the line through v . It can be longer than v , and if the angle between u and v is more than 90° (so $u \cdot v < 0$), it points opposite to v . See Figure 5.3.

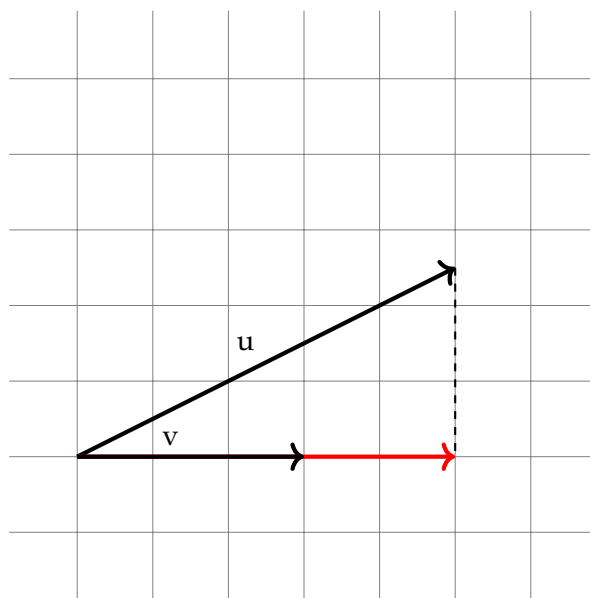


Figure 5.3: Projection vector extended beyond v .

To calculate the projection of $\mathbf{u}$ onto $\mathbf{v}$, use this formula:

$$\text{Projection of } \mathbf{u} \text{ onto } \mathbf{v}: \quad \text{proj}_{\mathbf{v}}(\mathbf{u}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|^2} \mathbf{v} = \frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \mathbf{v}$$

where $\mathbf{u} \cdot \mathbf{v}$ is the dot product of $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$. Note that either form of the equation works, and the $\mathbf{v}$ being multiplied by the dot product quotient cannot be cancelled because it is a vector, not a scalar.

Note that the denominator is the magnitude squared of vector $\mathbf{v}$.

$$\left(\sqrt{a_1^2 + a_2^2 + \dots + a_n^2} \right)^2$$

You learned previously that this is the same as the dot product of a vector with itself.

$$\mathbf{v} \cdot \mathbf{v}$$

In the examples that follow, we will simplify to the dot product notation.

Worked example: Let us find the projection of $\mathbf{u}$ onto $\mathbf{v}$ for a specific pair of vectors:

$$\mathbf{u} = (1, 4, 6)$$

$$\mathbf{v} = (-2, 6, 2)$$

$$\begin{aligned}\mathbf{proj}_v(\mathbf{u}) &= \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{v}\|^2} \mathbf{v} \\ \mathbf{proj}_v(\mathbf{u}) &= \frac{(1, 4, 6) \cdot (-2, 6, 2)}{(-2, 6, 2) \cdot (-2, 6, 2)} (-2, 6, 2) \\ \mathbf{proj}_v(\mathbf{u}) &= \frac{34}{44} (-2, 6, 2) \\ \mathbf{proj}_v(\mathbf{u}) &= (-1.545, 4.64, 1.545)\end{aligned}$$

As you work your way through this course, you will have a chance to apply the calculations you learn in this chapter to a variety of problems. Specifically, the Gram-Schmidt chapter shows how to transform a set of linearly independent vectors into a set of orthogonal ones. Projections are essential to that transformation.

Exercise 15 Projections

Find the projection of $\mathbf{a}$ on $\mathbf{b}$ where:

$$\mathbf{a} = (1, 3)$$

$$\mathbf{b} = (-4, 6)$$

Working Space

Answer on Page 79

Exercise 16 Projection onto an axis

Find the projection of $\mathbf{u} = (3, 4)$ onto $\mathbf{v} = (1, 0)$. Then find the orthogonal component, $\mathbf{u} - \mathbf{proj}_v(\mathbf{u})$.

Working Space

Answer on Page 80

Exercise 17 Sliding Down a Ramp

A skateboarder is on a ramp that drops 3

Working Space

m for every 4 m forward, so the downhill direction is $\mathbf{r} = (4, -3)$. Gravity accelerates them at $\mathbf{g} = (0, -9.8)$ m/s². Find the part of $\mathbf{g}$ that points along the ramp, and its size.

Answer on Page 80

5.1 Projections in Python

Create a file called `projections.py` and enter this code:

```
1 import numpy as np
2
3 # define two vectors
4 a = np.array([1, 4, 6])
5 b = np.array([-2, 6, 2])
6
7 # use np.dot() to calculate the dot product
8 projection_a_on_b = (np.dot(a, b)/np.dot(b, b))*b
9
10 print("The projection of vector a on vector b is:", projection_a_on_b)
```

You should see:

```
1 The projection of vector a on vector b is: [-1.54545455  4.63636364  1.54545455]
```

This matches the worked example.

5.2 Where to Learn More

Watch this Introduction to Projections from Khan Academy from your digital resources:
<https://rb.gy/yf0i3>

The Gram-Schmidt Process

Street grids are easy to navigate because the streets meet at right angles: going three blocks east never changes how far north you are. Directions that are perpendicular do not interfere with each other. The Gram-Schmidt Process takes any set of linearly independent vectors, however tilted, and straightens them into a set of orthogonal (perpendicular) vectors. The original vectors and the transformed vectors span the same subspace.

The process was named after two mathematicians: Jørgen Pedersen Gram, a Danish actuary mathematician, and Erhard Schmidt, a German mathematician. The men developed the orthogonalization process independently. Gram introduced the process in 1883, whereas Schmidt did his work in 1907. It was not named the Gram-Schmidt process until sometime later, after both mathematicians became well known in the mathematical community.

In the last chapter, you learned how to calculate a projection of one vector on another. Given two vectors, u and v , the projection separates u into the part that is orthogonal to v and the part that lies along v . The Gram-Schmidt Process builds on the notion of projections to iteratively strip away dependencies between vectors until the vectors that remain are orthogonal. If there happens to be a vector that is a linear combination of the others, that vector will be reduced to the zero vector. The vectors that remain define a new basis for space spanned by the original set of vectors.

Gram-Schmidt has many practical applications in science and engineering, such as:

1. In numerical computing, it helps solve least-squares problems, such as fitting a line to noisy data points.
2. In statistics and data analysis, it can reduce the complexity of a dataset so that it is easier to see which aspects or features contribute to the analysis.

6.1 The Process

The Gram-Schmidt Process orthogonalizes a set of vectors in an inner product space, most commonly the Euclidean space $\mathbb{R}^n$. The process takes a finite, linearly independent set $S = \{v_1, v_2, \dots, v_k\}$ for $k \leq n$, and generates an orthogonal set $S' = \{u_1, u_2, \dots, u_k\}$ that spans the same k -dimensional subspace of $\mathbb{R}^n$ as S .

Let us look at how the process works. Given a set of vectors $S = \{v_1, v_2, \dots, v_k\}$, the Gram-Schmidt Process is as follows:

1. Let $u_1 = v_1$.
2. For $j = 2, 3, \dots, k$:
 - (a) Let $w_j = v_j - \sum_{i=1}^{j-1} \frac{\langle v_j, u_i \rangle}{\langle u_i, u_i \rangle} u_i$
 - (b) Let $u_j = w_j$

Here, $\langle \cdot, \cdot \rangle$ denotes the inner product.

The set of vectors $S' = \{u_1, u_2, \dots, u_k\}$ obtained from this process is orthogonal, but not necessarily orthonormal. To create an orthonormal set, you simply need to normalize each vector u_i to unit length. That is, $u'_i = \frac{u_i}{\|u_i\|}$, where $\|\cdot\|$ denotes the norm (or length) of a vector.

Among other things, making vectors orthonormal simplifies calculations, makes it easier to define rotations and transformations, and provides a framework for calculations in fields such as quantum mechanics.

6.2 Example Calculation

Given a set of linearly independent vectors, we will use the Gram-Schmidt Process to find an orthogonal basis.

Let

$$W = \text{Span}(x_1, x_2, x_3)$$

where

$$x_1 = (1, 2, -2)$$

$$x_2 = (1, 0, -4)$$

$$x_3 = (5, 2, 0)$$

The three orthogonal vectors will define the same subspace as the original vectors.

The first vector of the orthogonal subspace is easy to define. We set it to be the same as x_1 .

$$v_1 = x_1 = (1, 2, -2)$$

For the second vector, take x_2 and subtract its projection onto v_1 . What is left is the part of x_2 perpendicular to v_1 :

$$v_2 = x_2 - \frac{x_2 \cdot v_1}{v_1 \cdot v_1} v_1$$

Substitute the values:

$$v_2 = (1, 0, -4) - \frac{(1, 0, -4) \cdot (1, 2, -2)}{(1, 2, -2) \cdot (1, 2, -2)} (1, 2, -2)$$

Calculate the coefficient for v_1 :

$$v_2 = (1, 0, -4) - \frac{9}{9}(1, 2, -2)$$

Perform the subtraction:

$$v_2 = (0, -2, -2)$$

For the third vector, subtract from x_3 its projections onto both v_1 and v_2 :

$$v_3 = x_3 - \frac{x_3 \cdot v_1}{v_1 \cdot v_1} v_1 - \frac{x_3 \cdot v_2}{v_2 \cdot v_2} v_2$$

Substitute the values:

$$v_3 = (5, 2, 0) - \frac{(5, 2, 0) \cdot (1, 2, -2)}{(1, 2, -2) \cdot (1, 2, -2)} (1, 2, -2) - \frac{(5, 2, 0) \cdot (0, -2, -2)}{(0, -2, -2) \cdot (0, -2, -2)} (0, -2, -2)$$

$$v_3 = (5, 2, 0) - (9/9)(1, 2, -2) - (-4/8)(0, -2, -2)$$

$$v_3 = (5, 2, 0) - (1, 2, -2) + (1/2)(0, -2, -2)$$

$$v_3 = (5, 2, 0) - (1, 2, -2) + (0, -1, -1)$$

$$v_3 = (4, -1, 1)$$

Check: $v_1 \cdot v_3 = 4 - 2 - 2 = 0$ and $v_2 \cdot v_3 = 0 + 2 - 2 = 0$. The set is orthogonal. To make it orthonormal, divide each vector by its length, $\frac{v}{\|v\|}$:

$$v_1 = (0.33, 0.67, -0.67)$$

$$v_2 = (0.0, -0.71, -0.71)$$

$$v_3 = (0.94, -0.24, 0.24)$$

Exercise 18 Gram-Schmidt Process

Use the Gram-Schmidt Process to find

Working Space

an orthogonal basis for the span defined by x_1, x_2 where:

$$x_1 = (1, 1, 1)$$

$$x_2 = (0, 1, 1)$$

Answer on Page 80

Exercise 19 Gram-Schmidt with Two Vectors

Apply the Gram-Schmidt Process to $x_1 = (1, 0, 1)$ and $x_2 = (1, 1, 0)$. Check that your two vectors are orthogonal.

Working Space

Answer on Page 81

6.3 The Gram-Schmidt Process in Python

Create a file called `vectors_gram-schmidt.py` and enter this code:

```

1  # import numpy to perform operations on vector
2  import numpy as np
3
4  # Find an orthogonal basis for the span of these three vectors
5  x1 = np.array([1, 2, -2])
6  x2 = np.array([1, 0, -4])
7  x3 = np.array([5, 2, 0])
8
9  # v1 = x1
10 v1 = x1
11 print("v1 = ", v1)
12
13 # v2 = x2 - (the projection of x2 on v1)
14 v2 = x2 - (np.dot(x2, v1)/np.dot(v1, v1))*v1
15 print("v2 = ", v2)
16
17 # v3 = x3 - (the projection of x3 on v1) - (the projection of x3 on v2)
18 v3 = x3 - (np.dot(x3, v1)/np.dot(v1, v1))*v1 - (np.dot(x3, v2)/np.dot(v2, v2))*v2

```

```
19 print("v3 =", v3)
20
21 # Next, normalize each vector to get a set of vectors that is both orthogonal and
   ⇨ orthonormal:
22 v1_norm = v1 / np.sqrt(np.sum(v1**2))
23 v2_norm = v2 / np.sqrt(np.sum(v2**2))
24 v3_norm = v3 / np.sqrt(np.sum(v3**2))
25 print("v1_norm =", v1_norm)
26 print("v2_norm =", v2_norm)
27 print("v3_norm =", v3_norm)
```

You should see:

```
1 v1 = [ 1  2 -2]
2 v2 = [ 0. -2. -2.]
3 v3 = [ 4. -1.  1.]
4 v1_norm = [ 0.33333333  0.66666667 -0.66666667]
5 v2_norm = [ 0.          -0.70710678 -0.70710678]
6 v3_norm = [ 0.94280904 -0.23570226  0.23570226]
```

6.4 Where to Learn More

Watch this video from Khan Academy about the Gram-Schmidt Process: <https://www.youtube.com/watch?v=rHonltF77zI>

Singular Value Decomposition

A photo on your phone is a big matrix of numbers. Netflix's record of who watched what is an even bigger one. Both are full of patterns, and the Singular Value Decomposition (SVD) is the tool that finds them. It is how images get compressed, how recommendation engines guess what you will like, and how scientists find the few important directions hidden in huge data sets.

You learned to find eigenvalues and eigenvectors, but only square matrices have them, and not every square matrix has real ones. Every matrix, of any shape, has singular values and singular vectors instead.

Singular Value Decomposition (SVD) is a matrix factorization technique that breaks down a matrix into three matrices that represent the structure and properties of the original matrix. The decomposed matrices make calculations easier and provide insight into the original matrix. SVD tells us that a linear transformation can be thought of as a rotation, scaling, and another rotation.

7.1 Definition

For any $m \times n$ matrix A , SVD decomposes the matrix into three matrices.

$$A = U\Sigma V^T \tag{7.1}$$

- U is an orthogonal matrix whose size is $m \times m$. Its columns are the eigenvectors of AA^T . These are the left singular vectors of A . Because U is orthogonal, $U^T U = I$.
- V is an orthogonal matrix whose size is $n \times n$ matrix. Its columns are the eigenvectors of $A^T A$. These are the right singular vectors of A . Because V is orthogonal, $V^T V = I$.
- Σ is a diagonal matrix that is the same size as A . Its diagonal contains the singular values of A , arranged in descending order. These values are the square roots of the eigenvalues of both $A^T A$ and AA^T .

7.2 Applications of SVD

SVD has numerous applications:

- It is used in machine learning and data science to perform dimensionality reduction, particularly through a technique known as Principal Component Analysis (PCA).
- In numerical linear algebra, SVD is used to solve linear equations and compute matrix inverses in a more numerically stable way.
- It is used in image compression, where low-rank approximations of an image matrix provide a compressed version of the original image.

7.3 Calculating SVD Manually

You might be inclined to skip this example because the computations are lengthy. Why would anyone do this when they can use a computing language, like Python, to calculate the SVD with essentially one command? We show this so you can understand what goes on “under the hood” when you compute SVD programmatically.

After you read through this example, you will see how to use Python to compute SVD. Next, you will see an example of using SVD for image compression. Finally, you will be given an exercise to compute the SVD. For this, you will need to write your own Python script.

Let us find the SVD for matrix A . Recall that we want to find U , Σ , and V^T such that:

$$A = U\Sigma V^T \quad (7.2)$$

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

The columns of U are the eigenvectors of AA^T , so we start by computing AA^T , a square matrix:

$$A^T = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

Next, we will find the eigenvalues and eigenvectors of AA^T . This is a chance to apply what you learned in the previous chapter. We know that:

$$Av = \lambda v \quad (7.3)$$

So:

$$\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Rewrite as a set of equations:

$$11x_1 + x_2 = \lambda x_1$$

$$x_1 + 11x_2 = \lambda x_2$$

Then rearrange:

$$(11 - \lambda)x_1 + x_2 = 0$$

$$x_1 + (11 - \lambda)x_2 = 0$$

This system has a nonzero solution only when the determinant is zero. Solve for λ :

$$\begin{vmatrix} 11 - \lambda & 1 \\ 1 & 11 - \lambda \end{vmatrix} = 0$$

And as equations:

$$(11 - \lambda)(11 - \lambda) - 1 \cdot 1 = 0$$

$$(\lambda - 10)(\lambda - 12) = 0$$

These are the eigenvalues.

$$\lambda = 10$$

$$\lambda = 12$$

When substituted into the original equations, you get the eigenvectors. For $\lambda = 10$:

$$(11 - 10)x_1 + x_2 = 0$$

$$x_1 = -x_2$$

We will set x_1 to 1 and get this eigenvector:

$$[1, -1]$$

For $\lambda = 12$:

$$(11 - 12)x_1 + x_2 = 0$$

$$x_1 = x_2$$

We will set x_1 to 1 and get this eigenvector:

$$[1, 1]$$

The matrix is:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Next, normalize the eigenvectors. They are already orthogonal, so Gram-Schmidt only needs to scale each one to length 1. After that, you will have U , the $m \times m$ matrix whose columns are eigenvectors of AA^T . These are the left singular vectors of A . After you apply

Gram-Schmidt, you should end up with:

$$U = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

The process for calculating V is the same as the calculation for U , except that we use the eigenvectors of $A^T A$:

$$A^T A = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

Its eigenvalues are 12, 10, and 0, with eigenvectors $[1, 2, 1]$, $[2, -1, 0]$, and $[1, 2, -5]$. (Check: $A^T A[1, 2, 1] = [12, 24, 12]$.) Normalizing them, in order of decreasing eigenvalue, gives:

$$V = \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{5} & 1/\sqrt{30} \\ 2/\sqrt{6} & -1/\sqrt{5} & 2/\sqrt{30} \\ 1/\sqrt{6} & 0 & -5/\sqrt{30} \end{bmatrix}$$

However, you want V^T :

$$V^T = \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6} \\ 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{30} & 2/\sqrt{30} & -5/\sqrt{30} \end{bmatrix}$$

You have only to calculate Σ , a diagonal matrix that is the same size as A . The diagonal contains the singular values of A , arranged in descending order. These are the square roots of the eigenvalues of both $A^T A$ and AA^T .

Because the nonzero eigenvalues of AA^T and $A^T A$ are the same, we use the eigenvalues 10 and 12 that we found for AA^T . Note that Σ will not be of the correct dimension to reconstruct the original matrix unless we add a column. By adding a zero column you will be able to multiply between U and V :

$$\Sigma = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

You can check your work by multiplying the decomposed matrices. This should return the original matrix.

$$\begin{aligned} A &= U\Sigma V^T \\ &= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6} \\ 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{30} & 2/\sqrt{30} & -5/\sqrt{30} \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{12}/\sqrt{2} & \sqrt{10}/\sqrt{2} & 0 \\ \sqrt{12}/\sqrt{2} & -\sqrt{10}/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6} \\ 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{30} & 2/\sqrt{30} & -5/\sqrt{30} \end{bmatrix} \end{aligned}$$

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

7.4 Singular Value Decomposition with Python

NumPy computes the SVD with one command, `np.linalg.svd`. Create `decomposition.py`:

```

1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4],
5               [5, 6]])
6
7 U, S, VT = np.linalg.svd(A)
8 print("U (left singular vectors):")
9 print(U)
10 print("Singular values:", S)
11 print("V transpose (right singular vectors):")
12 print(VT)
13
14 # Rebuild A. Sigma must be the same shape as A (3 x 2).
15 Sigma = np.zeros(A.shape)
16 Sigma[:, :2] = np.diag(S)
17 print("Rebuilt A:")
18 print(U @ Sigma @ VT)

```

You should see:

```

1 U (left singular vectors):
2 [[-0.2298477  0.88346102  0.40824829]
3  [-0.52474482  0.24078249 -0.81649658]
4  [-0.81964194 -0.40189603  0.40824829]]
5 Singular values: [9.52551809 0.51430058]
6 V transpose (right singular vectors):
7 [[-0.61962948 -0.78489445]
8  [-0.78489445  0.61962948]]
9 Rebuilt A:
10 [[1. 2.]
11  [3. 4.]
12  [5. 6.]]

```

NumPy returns the singular values as a list, so we put them into a matrix Σ the same shape as A before multiplying.

7.5 Sign Ambiguity

You might notice that at times, the absolute values in the U and V^T matrices are correct, but that the signs vary from what you see as the answer. For example, when you compare a manually calculated SVD with one done in Python the signs might not agree. Both decompositions of A are valid. Both decompositions will satisfy:

$$A = U\Sigma V^T$$

The singular values on the diagonal of Σ are always nonnegative, so they never have this ambiguity.

The sign ambiguity has implications. For example, when using SVD to compress data, if some of the signs are flipped, the data can have artifacts. At this point in your education, you do not need to concern yourself with it, except when you are comparing SVD results for the same matrix.

Exercise 20 Singular Value Decomposition

Modify your Python code to calculate SVD for the matrix in the worked out example. Did you arrive at the same answer? Keep in mind that Python will compute square roots and present fractions as decimal. Take a look at the signs for the values in the U and V^T matrices. Are they the same, or is this an example of sign ambiguity?

Working Space

Answer on Page 81

Exercise 21 Singular Values of a Simple Matrix

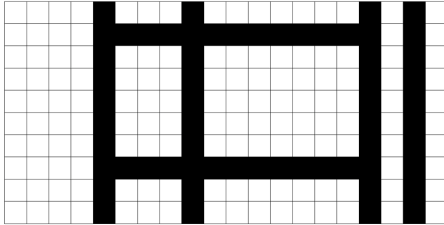
Find the singular values of $B = \begin{bmatrix} 0 & 3 \\ 0 & 4 \end{bmatrix}$. Start by finding the eigenvalues of BB^T . Then take their square roots.

Working Space

Answer on Page 81

7.6 SVD Applied to Image Compression

This image is a grid of 20 pixels across and 10 pixels down, each of which is either black or white.



It is a simple image that has only two types of rows, which makes it ideal for data compression. A row is either the first pattern or the second. Think of a striped shirt. Once you know the stripe pattern, you only need to describe the stripes, not every thread.



We can represent the data as a 10 by 20 matrix whose 200 entries are either 0 for black or 1 for white.

```

[00001000100000001010
00001111111111111010
00001000100000001010
00001000100000001010
00001000100000001010
00001000100000001010
00001000100000001010
00001000100000001010
00001111111111111010
00001000100000001010
00001000100000001010]

```

When you perform an SVD on this matrix, there are only two non-zero singular values, 6.79 and 3.72. (Try it in Python.) Thus, you can represent the matrix as:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T$$

This means there are two $\mathbf{u}$ vectors, each with 10 entries, two $\mathbf{v}$ vectors each with 20 entries, and two singular values. Add those up: $2 \cdot 10 + 2 \cdot 20 + 2 = 62$. This implies that the image can be represented by 62 values instead of 200. If you look back at the image, you can see that there are many dependent rows and very few independent ones.

This is a simple image and a small pixel matrix, but it should give you a sense of how SVD can decompose an image in a way that identifies how much of the image is redundant, and therefore can be compressed.

7.7 Where to Learn More

We Recommend a Singular Value Decomposition. This American Mathematical Society publication focuses the geometry of SVD. What we like about the article is that it shows both graphically and numerically how SVD can be used for data compression on images and for noise reduction. The data compression example in your workbook is based on this article. <https://www.ams.org/publicoutreach/feature-column/fcarc-svd>

Sign Ambiguity in Singular Value Decomposition (SVD). This is a good article for those who want a deeper understanding of sign ambiguity. <https://www.educative.io/blog/sign-ambiguity-in-singular-value-decomposition>

Singular Value Decomposition Tutorial. This PDF starts by defining points, space, and vectors and works through all the concepts you need to tackle SVD. It is one of the few resources that has a completely worked out example of manually calculating SVD. The example in this chapter is from that tutorial. If you read the entire paper, you will find that it is a good review of the concepts you have studied in previous chapters. <https://rb.gy/j6s0w>

Answers to Exercises

Answer to Exercise 1 (on page 5)

$$(1 - 2, 2 + 1, 3 - 4) = (-1, 3, -1).$$

Answer to Exercise 2 (on page 5)

The normal $\langle 1, 1, 1 \rangle$ works, because it is perpendicular to both $\langle -1, 1, 0 \rangle$ and $\langle -1, 0, 1 \rangle$ (the vectors from $(1, 0, 0)$ to the other two points). Using the point $(1, 0, 0)$, $d = 1$, so the plane is $x + y + z = 1$.

Answer to Exercise 3 (on page 5)

$$D = \frac{|0-1|}{\sqrt{3}} = \frac{1}{\sqrt{3}} = \frac{\sqrt{3}}{3} \approx 0.577.$$

Answer to Exercise 4 (on page 5)

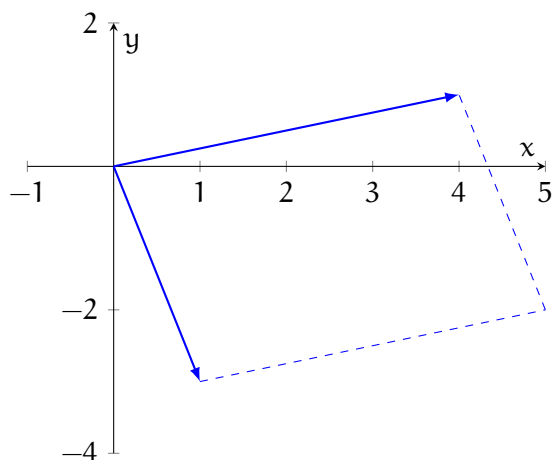
Substituting gives $(2 + t) + (-1 + 3t) + (4 - t) = 8$, so $5 + 3t = 8$ and $t = 1$. The point is $(3, 2, 3)$.

Answer to Exercise 5 (on page 6)

Substituting gives $t + (1 + t) + (-2t) = 5$, which simplifies to $1 = 5$. That is never true, so there is no meeting point: the line is parallel to the plane. (Check: the direction $\langle 1, 1, -2 \rangle$ dotted with the normal $\langle 1, 1, 1 \rangle$ is 0, so the line runs perpendicular to the normal, which means parallel to the plane.)

Answer to Exercise 6 (on page 10)

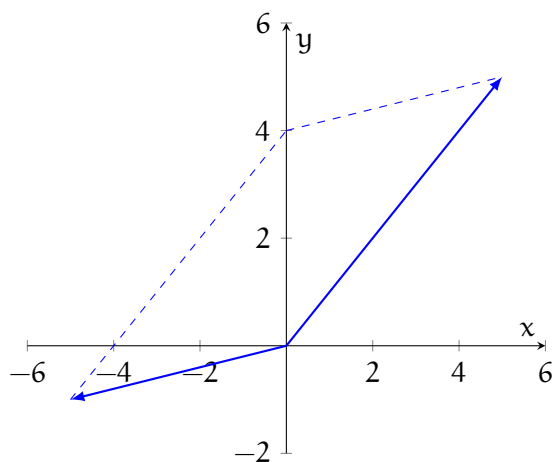
1. Our two vectors from the columns of the matrix are $[1, -3]$ and $[4, 1]$. Plotting:



The area of this parallelogram is the same as the determinant of the matrix:

$$\det \left(\begin{bmatrix} 1 & 4 \\ -3 & 1 \end{bmatrix} \right) = 1 \cdot 1 - (4 \cdot -3) = 1 + 12 = 13$$

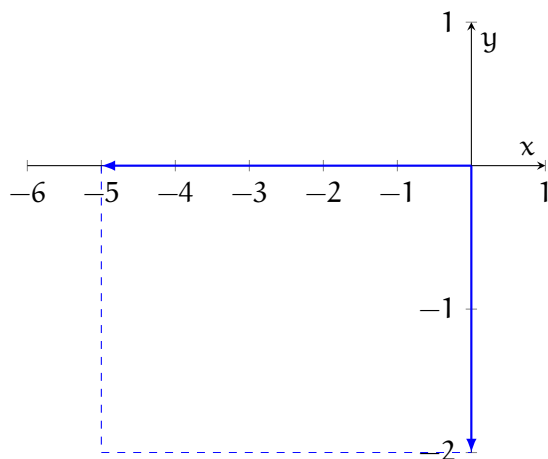
2. Our two vectors from the columns of the matrix are $[5, 5]$ and $[-5, -1]$. Plotting:



The area of this parallelogram is the same as the determinant of the matrix:

$$\det \left(\begin{bmatrix} 5 & -5 \\ 5 & -1 \end{bmatrix} \right) = 5 \cdot -1 - (-5 \cdot 5) = -5 + 25 = 20$$

3. Our two vectors from the columns of the matrix are $[0, -2]$ and $[-5, 0]$. Plotting:



This is a rectangle, and we can see the area is $5 \cdot 2 = 10$. However, the determinant is:

$$\det \left(\begin{bmatrix} 0 & -5 \\ -2 & 0 \end{bmatrix} \right) = 0 \cdot 0 - (-5 \cdot -2) = 0 - 10 = -10$$

The sign is negative because the second column is a clockwise turn from the first, so the orientation is reversed. The area is $|-10| = 10$. We will discuss orientation in a future chapter.

Answer to Exercise 7 (on page 16)

First, find the determinant of A :

$$\det(A) = (1)(-1) - (2)(0) = -1$$

So our inverse is given by:

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

Surprise: $A^{-1} = A$. Check: $AA = I$. Applying this matrix twice gets you back where you started.

Answer to Exercise 8 (on page 16)

First, check that A has an inverse:

$$\det A = 1 \begin{vmatrix} 1 & 1 \\ 4 & -3 \end{vmatrix} - 2 \begin{vmatrix} 0 & 1 \\ 2 & -3 \end{vmatrix} - 2 \begin{vmatrix} 0 & 1 \\ 2 & 4 \end{vmatrix} = 1(-3 - 4) - 2(0 - 2) - 2(0 - 2) = -7 + 4 + 4 = 1$$

Since $\det(A) \neq 0$, A must have an inverse. We find the inverse with the augmented matrix method:

$$\begin{aligned}
 \left[\begin{array}{ccc|ccc} 1 & 2 & -2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 2 & 4 & -3 & 0 & 0 & 1 \end{array} \right] & \xrightarrow{R_3 \leftarrow R_3 - 2R_1} \left[\begin{array}{ccc|ccc} 1 & 2 & -2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -2 & 0 & 1 \end{array} \right] \\
 & \xrightarrow{R_2 \leftarrow R_2 - R_3} \left[\begin{array}{ccc|ccc} 1 & 2 & -2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & -1 \\ 0 & 0 & 1 & -2 & 0 & 1 \end{array} \right] \\
 & \xrightarrow{R_1 \leftarrow R_1 + 2R_3} \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -3 & 0 & 2 \\ 0 & 1 & 0 & 2 & 1 & -1 \\ 0 & 0 & 1 & -2 & 0 & 1 \end{array} \right] \\
 & \xrightarrow{R_1 \leftarrow R_1 - 2R_2} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -7 & -2 & 4 \\ 0 & 1 & 0 & 2 & 1 & -1 \\ 0 & 0 & 1 & -2 & 0 & 1 \end{array} \right].
 \end{aligned}$$

This gives us our inverse on the right side, $A^{-1} = \begin{bmatrix} -7 & -2 & 4 \\ 2 & 1 & -1 \\ -2 & 0 & 1 \end{bmatrix}$

To check our answer, we can use $AA^{-1} = I_3$:

$$\begin{aligned}
 AA^{-1} &= \begin{bmatrix} 1 & 2 & -2 \\ 0 & 1 & 1 \\ 2 & 4 & -3 \end{bmatrix} \begin{bmatrix} -7 & -2 & 4 \\ 2 & 1 & -1 \\ -2 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1(-7) + 2(2) + (-2)(-2) & 1(-2) + 2(1) + (-2)(0) & 1(4) + 2(-1) + (-2)(1) \\ 0(-7) + 1(2) + 1(-2) & 0(-2) + 1(1) + 1(0) & 0(4) + 1(-1) + 1(1) \\ 2(-7) + 4(2) + (-3)(-2) & 2(-2) + 4(1) + (-3)(0) & 2(4) + 4(-1) + (-3)(1) \end{bmatrix} \\
 &= \begin{bmatrix} -7 + 4 + 4 & -2 + 2 + 0 & 4 - 2 - 2 \\ 0 + 2 - 2 & 0 + 1 + 0 & 0 - 1 + 1 \\ -14 + 8 + 6 & -4 + 4 + 0 & 8 - 4 - 3 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3.
 \end{aligned}$$

Answer to Exercise 9 (on page 17)

No. Each row is a scalar multiple of the first row:

- Row 2 = $2 \times$ Row 1
- Row 3 = $4 \times$ Row 1
- Row 4 = $8 \times$ Row 1

So the rows are *linearly dependent*, meaning the matrix has rank 1, and its determinant is 0. A matrix with determinant 0 is singular, so no inverse exists.

Answer to Exercise 10 (on page 35)

The matrix that represents this transformation is:

$$A = \begin{bmatrix} 1.5 & 0 \\ -2 & -1 \end{bmatrix}$$

We find this by first noticing that the house and door components are flipped over the x-axis, which means that the y-values of the points are flipped. This gives us a negative one value in the e_2 basis of the matrix.

Next we can see that the house is stretched horizontally in the x-direction. The orange dots end at 4.5, which is 1.5 times the original length of 3, so we know that the x-values are scaled by a factor of 1.5. This gives us the value of 1.5 in the first row of the matrix.

Finally, the roof is slanted, which means that there is a shearing component to this transformation. Notice that the roof (door corner) is moved up, so the e_1 also has a negative value y-component. This gives us the value of -2 in the second row, first column of the matrix.

Check with a corner: $A \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 4.5 \\ -6 - 3 \end{bmatrix} = \begin{bmatrix} 4.5 \\ -9 \end{bmatrix}$, which is the bottom-right orange point.

Answer to Exercise 11 (on page 39)

1. The original matrix for an x-shear with a shear factor of k is:

$$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$$

2. The inverse matrix for this transformation is:

$$\begin{bmatrix} 1 & -k \\ 0 & 1 \end{bmatrix}$$

(the determinant is 1, and the inverse is found by swapping the diagonal elements and negating the off-diagonal elements, which in this case results in negating k).

Answer to Exercise 12 (on page 40)

Reflecting sends $\vec{e}_1 \mapsto \vec{e}_1$ and $\vec{e}_2 \mapsto -\vec{e}_2$; doubling then gives $\vec{e}_1 \mapsto 2\vec{e}_1$ and $\vec{e}_2 \mapsto -2\vec{e}_2$. So the matrix is $\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$, and $(1, 2) \mapsto (2, -4)$. The determinant is -4 : areas grow by a factor of 4, and the negative sign says the shape has been flipped.

Answer to Exercise 13 (on page 48)

First, write the $A - \lambda I$ equation

$$\begin{bmatrix} 1 - \lambda & 2 \\ 2 & -2 - \lambda \end{bmatrix}$$

Our goal is to get the determinant of $A - \lambda I$ to be equal to zero. For a 2×2 , the determinant is $ad - bc$:

$$\begin{aligned} (1 - \lambda)(-2 - \lambda) - 4 &= 0 \\ (-2 + 2\lambda - \lambda + \lambda^2) - 4 &= 0 \\ \lambda^2 + \lambda - 6 &= 0 \\ \lambda_1 = 2, \quad \lambda_2 &= -3 \end{aligned}$$

Then we solve for the two eigenvectors, $\mathbf{e}_n$, using $A - \lambda I$

For $\lambda_1 = 2$:

$$\lambda_1 = 2 : A - 2I = \begin{bmatrix} -1 & 2 \\ 2 & -4 \end{bmatrix}$$

We solve the following using row-column multiplication (rows of numbers and columns of x_i)

$$\begin{bmatrix} -1 & 2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} -x_1 + 2x_2 = 0 \\ 2x_1 - 4x_2 = 0 \end{cases}$$

We notice, $x_1 = 2x_2$. This means every eigenvector is a multiple of $\mathbf{e}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ (using $x_2 = 1$).

For $\lambda_2 = -3$:

$$\lambda_2 = -3 : A + 3I = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} 4x_1 + 2x_2 = 0 \\ 2x_1 + x_2 = 0 \end{cases}$$

This results in $x_2 = -2x_1$, meaning $\mathbf{e}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$.

Answer to Exercise 14 (on page 50)

For an upper triangular matrix, the eigenvalues are the diagonal entries, $\lambda = 4$ and $\lambda = 2$.

For $\lambda = 4$: $A - 4I = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix}$, so $x_2 = 0$ and $\vec{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

For $\lambda = 2$: $A - 2I = \begin{bmatrix} 2 & 1 \\ 0 & 0 \end{bmatrix}$, so $2x_1 + x_2 = 0$ and $\vec{v}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$.

Check: $A\vec{v}_2 = \begin{bmatrix} 2 \\ -4 \end{bmatrix} = 2\vec{v}_2$.

Answer to Exercise 15 (on page 56)

Compute the dot product of $\mathbf{a}$ and $\mathbf{b}$:

$$1(-4) + 3(6) = -4 + 18 = 14$$

Compute the dot product of $\mathbf{b}$ with itself:

$$(-4)^2 + 6^2 = 16 + 36 = 52$$

Then the projection is

$$\frac{14}{52}(-4, 6) = \left(-\frac{14}{13}, \frac{21}{13}\right) \approx (-1.077, 1.615)$$

Answer to Exercise 16 (on page 56)

The dot product is $\mathbf{u} \cdot \mathbf{v} = 3$, and $\mathbf{v} \cdot \mathbf{v} = 1$. So

$$\text{proj}_{\mathbf{v}}(\mathbf{u}) = \frac{3}{1}(1, 0) = (3, 0)$$

The orthogonal component is $(3, 4) - (3, 0) = (0, 4)$. It is perpendicular to $\mathbf{v}$, since $(0, 4) \cdot (1, 0) = 0$.

Answer to Exercise 17 (on page 56)

$\mathbf{g} \cdot \mathbf{r} = 0 + 29.4 = 29.4$ and $\mathbf{r} \cdot \mathbf{r} = 16 + 9 = 25$, so

$$\text{proj}_{\mathbf{r}}(\mathbf{g}) = \frac{29.4}{25}(4, -3) = (4.704, -3.528) \text{ m/s}^2$$

Its length is $\sqrt{4.704^2 + 3.528^2} = 5.88 \text{ m/s}^2$, which is 0.6×9.8 . The ramp's slope angle θ has $\sin \theta = \frac{3}{5} = 0.6$, so this matches the physics formula $g \sin \theta$.

Answer to Exercise 18 (on page 61)

The first vector of the orthogonal subspace is:

$$\mathbf{v}_1 = \mathbf{x}_1 = (1, 1, 1)$$

The second vector is $\mathbf{x}_2$ minus its projection onto $\mathbf{v}_1$:

$$\mathbf{v}_2 = \mathbf{x}_2 - \frac{\mathbf{x}_2 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1} \mathbf{v}_1$$

Substitute the values:

$$\mathbf{v}_2 = (0, 1, 1) - \frac{(0, 1, 1) \cdot (1, 1, 1)}{(1, 1, 1) \cdot (1, 1, 1)}(1, 1, 1)$$

$$\mathbf{v}_2 = (0, 1, 1) - (2/3)(1, 1, 1)$$

$$\mathbf{v}_2 = (-2/3, 1/3, 1/3)$$

Normalize each vector by dividing by its length. Since $\|v_1\| = \sqrt{3}$, we get $v'_1 = \frac{1}{\sqrt{3}}(1, 1, 1) \approx (0.577, 0.577, 0.577)$. Since $\|v_2\| = \frac{\sqrt{6}}{3} \approx 0.816$, we get $v'_2 \approx (-0.816, 0.408, 0.408)$.

Answer to Exercise 19 (on page 62)

Let $v_1 = x_1 = (1, 0, 1)$. Then $x_2 \cdot v_1 = 1$ and $v_1 \cdot v_1 = 2$, so

$$v_2 = (1, 1, 0) - \frac{1}{2}(1, 0, 1) = \left(\frac{1}{2}, 1, -\frac{1}{2}\right)$$

Multiplying by 2 gives the same direction without fractions: $(1, 2, -1)$. Check: $(1, 0, 1) \cdot (1, 2, -1) = 1 - 1 = 0$.

Answer to Exercise 20 (on page 70)

$$U = \begin{bmatrix} -0.70710678 & -0.70710678 \\ -0.70710678 & 0.70710678 \end{bmatrix}$$

$$\text{Singular values} = [3.464101623, 1.6227766]$$

$$V^T = \begin{bmatrix} -0.408 & -0.816 & -0.408 \\ -0.894 & 0.447 & 0.0 \\ -0.183 & -0.365 & 0.9129 \end{bmatrix}$$

Answer to Exercise 21 (on page 70)

$$BB^T = \begin{bmatrix} 0 & 3 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 9 & 12 \\ 12 & 16 \end{bmatrix}$$

The characteristic equation is $\lambda^2 - 25\lambda = 0$, so $\lambda = 25$ or $\lambda = 0$. The singular values are $\sqrt{25} = 5$ and $\sqrt{0} = 0$. Check: the sum of the squares of the singular values equals the sum of the squares of all entries of B , which is $9 + 16 = 25$.



INDEX

- characteristic equation, [44](#)
- codomain, [21](#)
- cofactor expansion, [12](#)
- determinant, [7](#)
 - 2 by 2, [8](#)
 - expansion by minors, [11](#)
 - n by n, [11](#)
 - scaling, [8](#)
- direction vector, [3](#)
- distance
 - point to plane, [4](#)
- domain, [21](#)
- eigenbasis, [48](#)
- eigenequation, [44](#)
- eigenspace, [47](#)
- eigenvalue, [43](#)
 - calculating, [44](#)
- eigenvector, [43](#)
 - reflections, [50](#)
 - rotations, [50](#)
- Gram-Schmidt
 - applications, [59](#)
 - example, [60](#)
 - history, [59](#)
 - procedure, [59](#)
- Gram-Schmidt Process, [59](#)
- Gram-Schmidt process
 - python script for, [62](#)
- identity transformation, [22](#)
- image compression, [71](#)
- inverse
 - 2 by 2 formula, [15](#)
 - augmented matrix method, [16](#)
- inverse matrix, [13](#)
- inverse transformation, [38](#)
- invertible, [14](#)
- line
 - in space, [3](#)
- linear system
 - inverse solution, [17](#)
- low-rank approximation, [66](#)
- matrix transformation, [21](#)
- minor matrix, [12](#)
- non-invertible transformation, [39](#)
- normal vector, [3](#)
- numpy
 - eig, [49](#)
- orthogonal, [53](#)
- orthogonal basis, [60](#)
- orthogonal matrix, [65](#)
- orthonormal set, [60](#)
- parallelogram area, [7](#)
- parametric equations, [3](#)
- plane
 - in space, [3](#)

principal component analysis, 66

projection

 beyond the vector, 54

 example, 55

 orthogonal component, 53

 Python, 57

 shadow analogy, 54

projections, 53

 formula for, 55

 visualization of, 53

reflections, 29

rotation matrix, 36

Sarrus rule, 11

scaling, 24

shear, 24

sign ambiguity, 70

singular matrix, 14

singular value decomposition, 65

singular values, 65

singular vectors, 65

SVD

 manual calculation, 66

svd, 65

 python script for, 69

vector

 projection, 53

vector equation, 3

 line, 3