



SPECTRAL STRUCTURE OF OPERATORS IN HILBERT SPACE

Lecture 6

THE STORY SO FAR...

We began with complex numbers.

saw that complex amplitudes carry magnitude and phase, and that geometry in complex space requires conjugation.

introduced inner products to define length and angle.

defined the adjoint as the operator determined by inner product symmetry.

saw that unitary operators preserve that geometry.

Normalized vectors in $\mathbb{C}^2$ live on a 3-sphere.

EIGENVALUES AND EIGENVECTORS

Augustin-Louis Cauchy (1820s) studied what he called *characteristic values* of matrices.

David Hilbert, in developing what we now call Hilbert spaces, popularized the German terms **Eigenwert** and **Eigenvektor**.

Eigen is a German adjective meaning “own” or “intrinsic.” (“wert” and “vector” are value and vector)

For example:

Jeder Mensch hat seine eigene Geschichte. (Everyone has their own story.)

An eigenvalue and eigenvector are therefore the **intrinsic value** and **intrinsic direction** of an operator.

They are not imposed by coordinates.

They belong to the transformation itself.

For many operators in Hilbert space, especially Hermitian and unitary operators, this intrinsic data completely determines the operator.

*In many linear algebra courses, eigenvalues and eigenvectors are presented as computational tools for diagonalizing matrices and solving systems of equations. Here, we will treat them as revealing the intrinsic structure of operators in Hilbert space.

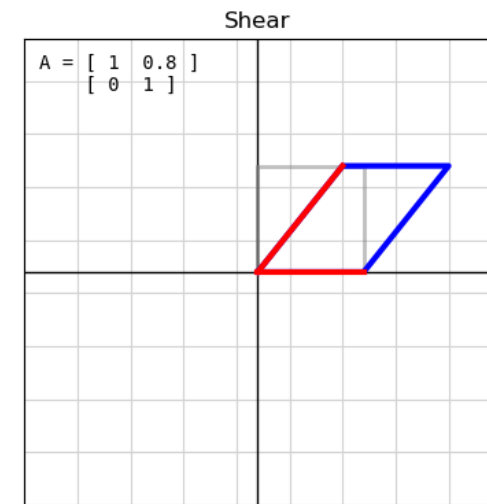
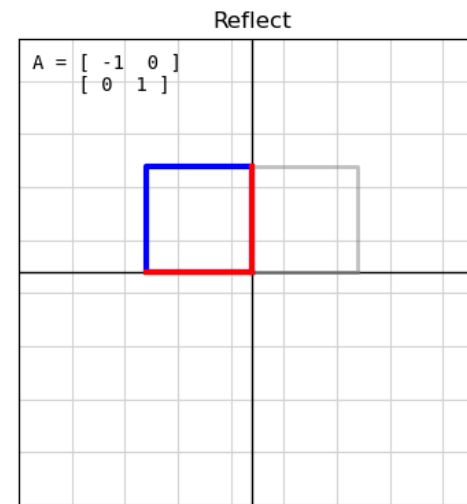
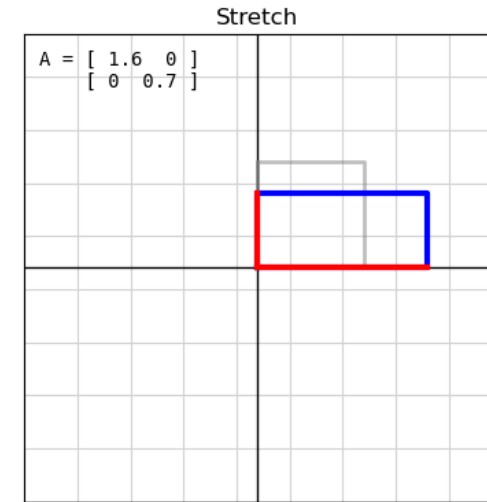
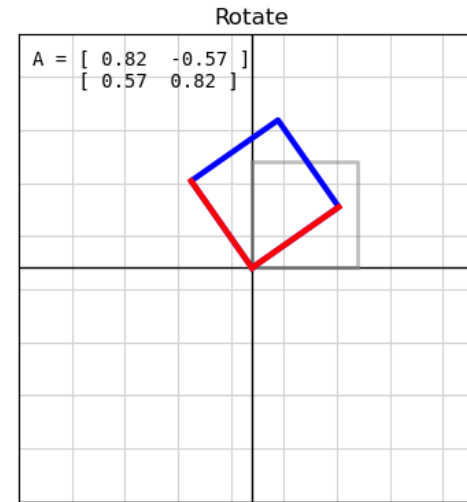
RECALL... MATRICES AS LINEAR MAPS

MULTIPLICATION AS COMPOSITION OF MAPS

Linear Maps: Geometry of 2×2 Matrices

A linear map can Rotate,
Stretch, Reflect, Shear

It will turn out that quantum
computing depends on linear
maps. Non-linear
transformations break quantum
computing.



EIGENVALUES AND EIGENVECTORS IN $\mathbb{R}^N$

Let A be a linear operator on a vector space.

A nonzero vector v is called an **eigenvector** of A if

$$A v = \lambda v$$

for some scalar λ .

The scalar λ is called the **eigenvalue** corresponding to v .

In other words linear operators rotate and scale vectors. But when applied to certain vectors some operators don't rotate the vector they just scale them.

When that happens the operator just acts like a scalar value multiplying the vector.

Those vectors that don't rotate are the Eigenvectors and the amount they scale is the Eigenvalue.

*In many linear algebra courses, eigenvalues and eigenvectors are presented as computational tools for diagonalizing matrices and solving systems of equations. Here, we will treat them as revealing the intrinsic structure of operators in Hilbert space.

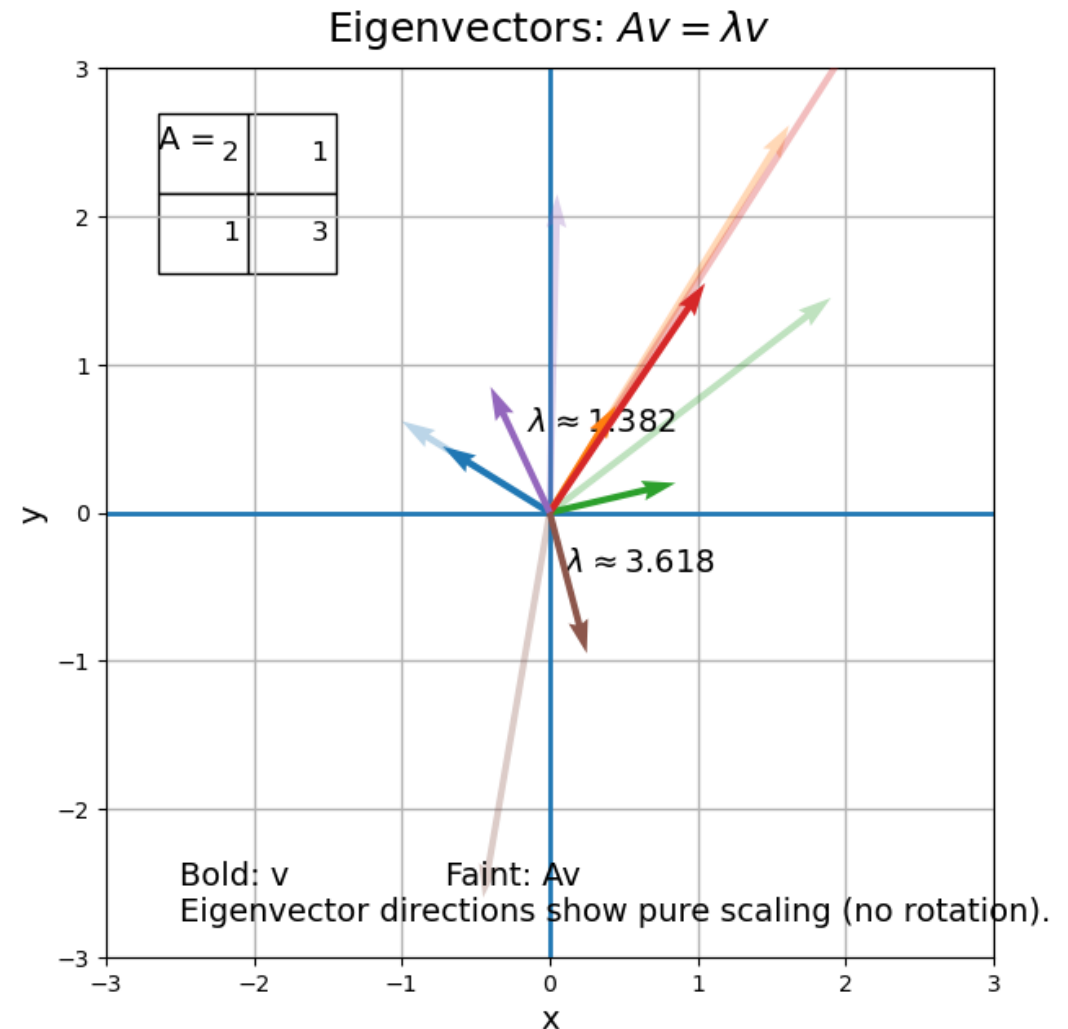
EIGENVALUES AND EIGENVECTORS IN $\mathbb{R}^N$

In other words linear operators rotate and scale vectors. But when applied to certain vectors some operators don't rotate the vector they just scale them.

When that happens the operator just acts like a scalar value multiplying the vector.

Those vectors that don't rotate are the Eigenvectors and the amount they scale is the Eigenvalue.

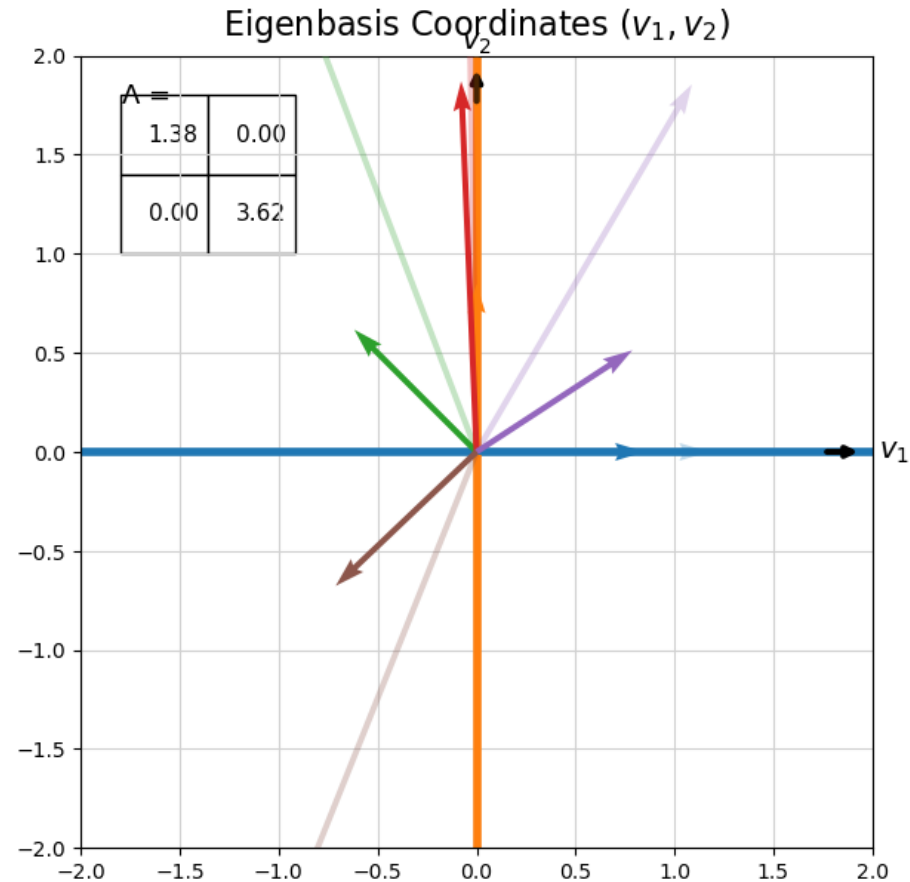
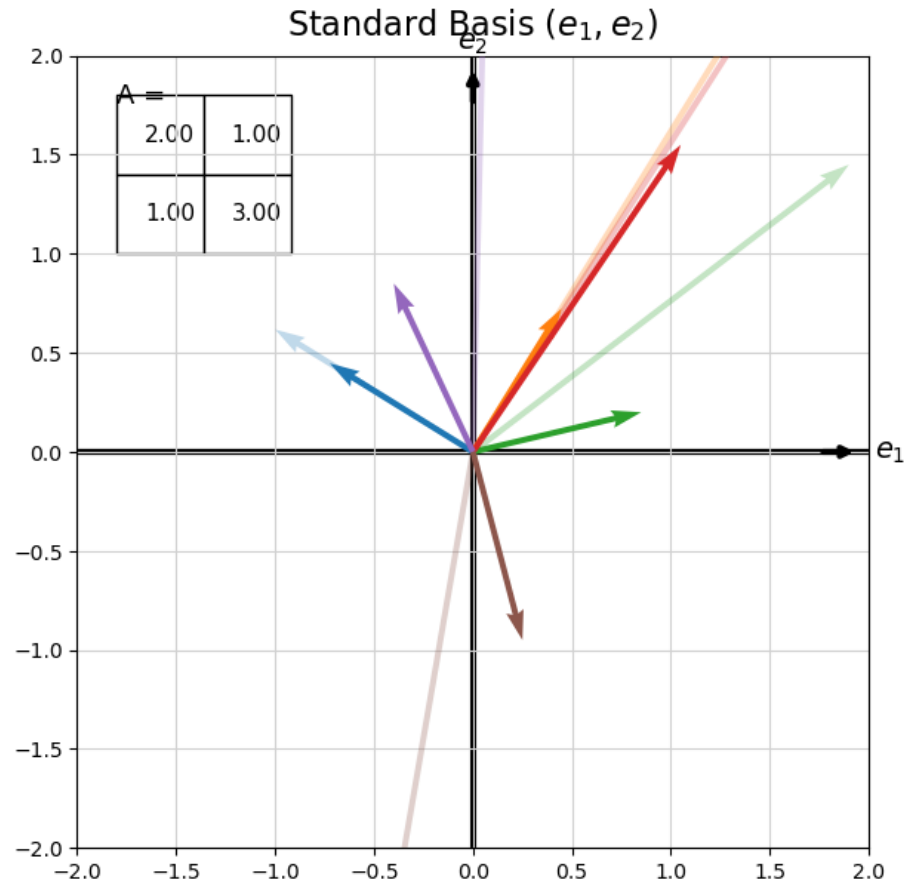
Notice how the blue and orange vectors only scale they don't rotate.



*In many linear algebra courses, eigenvalues and eigenvectors are presented as computational tools for diagonalizing matrices and solving systems of equations. Here, we will treat them as revealing the intrinsic structure of operators in Hilbert space.

EIGENVALUES AND EIGENVECTORS IN $\mathbb{R}^N$

Eigenbasis: Instead of using the standard axes $e_1 = (1,0)$, $e_2 = (0,1)$ (left panel) we choose axes aligned with the eigenvectors of A ; in this coordinate system the matrix becomes diagonal $\Lambda = \text{diag}(1.38, 3.62)$, so the operator no longer mixes directions and acts independently on each coordinate.



*In many linear algebra courses, eigenvalues and eigenvectors are presented as computational tools for diagonalizing matrices and solving systems of equations. Here, we will treat them as revealing the intrinsic structure of operators in Hilbert space.

WHY WE CARE ABOUT VECTORS THAT SCALE AND DON'T ROTATE WHEN AN OPERATOR IS APPLIED?

Eigenvectors reveal the **intrinsic structure of the operator**.

In the eigenvector basis:

- the operator does not mix directions
- each direction evolves independently
- the matrix becomes **diagonal**

Instead of

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

we get

$$A = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

The complicated operator decomposes into simple scalings.

Quantum Computing: Quantum states are often written in the eigenbasis of an operator because in that basis the operator does not mix directions, allowing each component of the state to evolve independently.

NOT ALL OPERATORS HAVE SUCH VECTORS IN $\mathbb{R}^N$

Example: a rotation by 90° $\left(\frac{\pi}{2}\right)$

$$R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

No non-zero vector satisfies

$$Rv = \lambda v$$

so R has no real eigenvectors.

NOT ALL OPERATORS HAVE SUCH VECTORS IN $\mathbb{R}^N$

Example: a rotation by 90° $\left(\frac{\pi}{2}\right)$

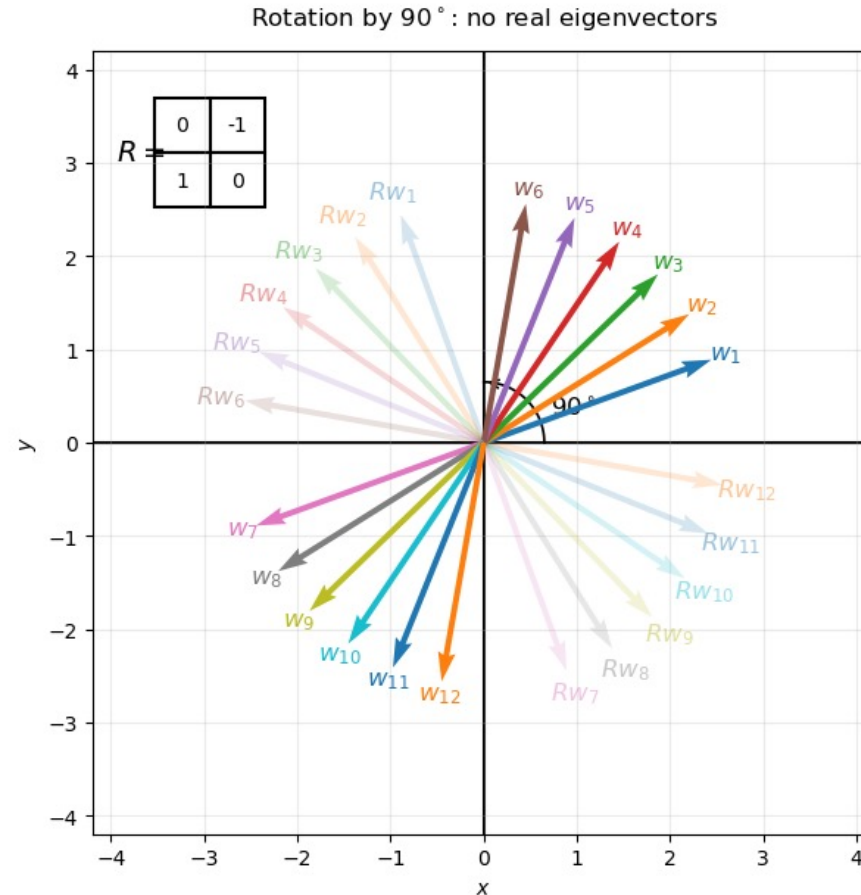
$$R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

No non-zero vector satisfies

$$Rw = \lambda w$$

so R has **no real eigenvectors**.

In the figure the input vectors w have images Rw . Since there is rotational symmetry there can be no eigenvectors. There are no *special directions*.



HOW TO CALCULATE EIGENVALUES AND EIGENVECTORS

By definition an eigenvector v satisfies

$$Av = \lambda v$$

Rearrange:

$$Av - \lambda v = 0$$

Factor:

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

For a non-zero vector v to exist,

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

The **determinant**, $\det(A)$, is a number that tells us whether a matrix is invertible; for a 2×2 matrix

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

and when the determinant is **zero**, the matrix collapses space in some direction.

HOW TO CALCULATE EIGENVALUES AND EIGENVECTORS

- Blue square has **area = 1**
- Red parallelogram has **area = $\det(A)$**
- For the matrix

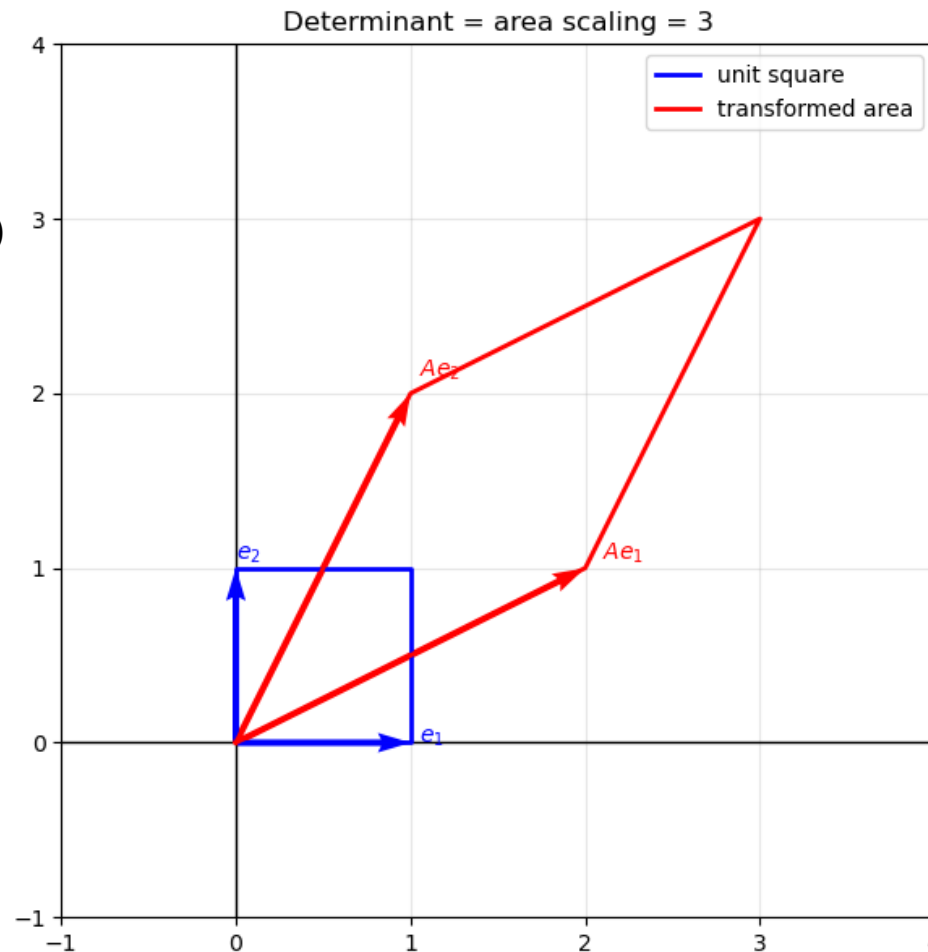
$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$$\det(A) = 3$$

so the area triples.

The area is positive for this matrix.

(We will use this in lecture 12)



Transforms the unit square into a parallelogram

$\det = 0 \rightarrow$ area = 0 (collapse)

$|\det| < 1 \rightarrow$ area shrinks

$|\det| > 1 \rightarrow$ area expands

$\det < 0 \rightarrow$ orientation flips (meaning the order of vertexes flips)

HOW TO CALCULATE EIGENVALUES AND EIGENVECTORS

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

Compute the determinant

$$\det(A) = (1)(4) - (2)(2)$$

$$= 4 - 4 = 0$$

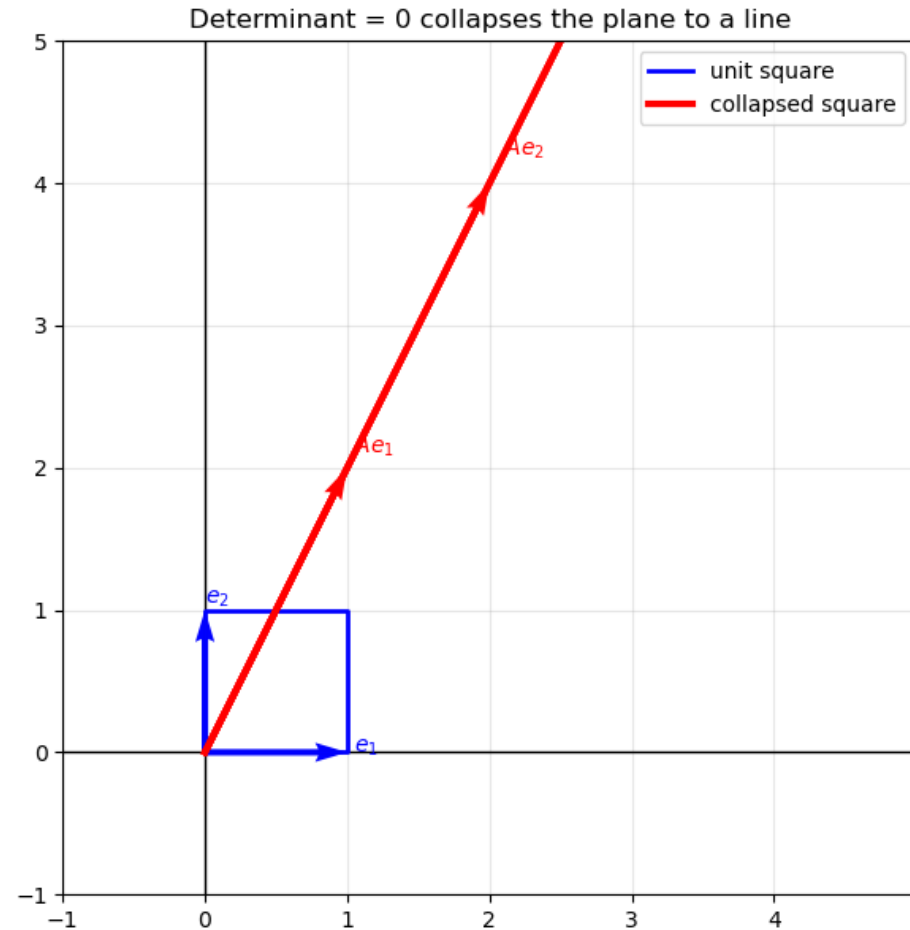
So

$$\det(A) = 0$$

Because $\det(A) = 0$, the transformation collapses area.

Vectors along the eigenvector with eigenvalue 0 collapse to the origin.

Vectors along the other eigenvector are stretched. As a result, the square collapses into a **line**.



HOW TO CALCULATE EIGENVALUES AND EIGENVECTORS

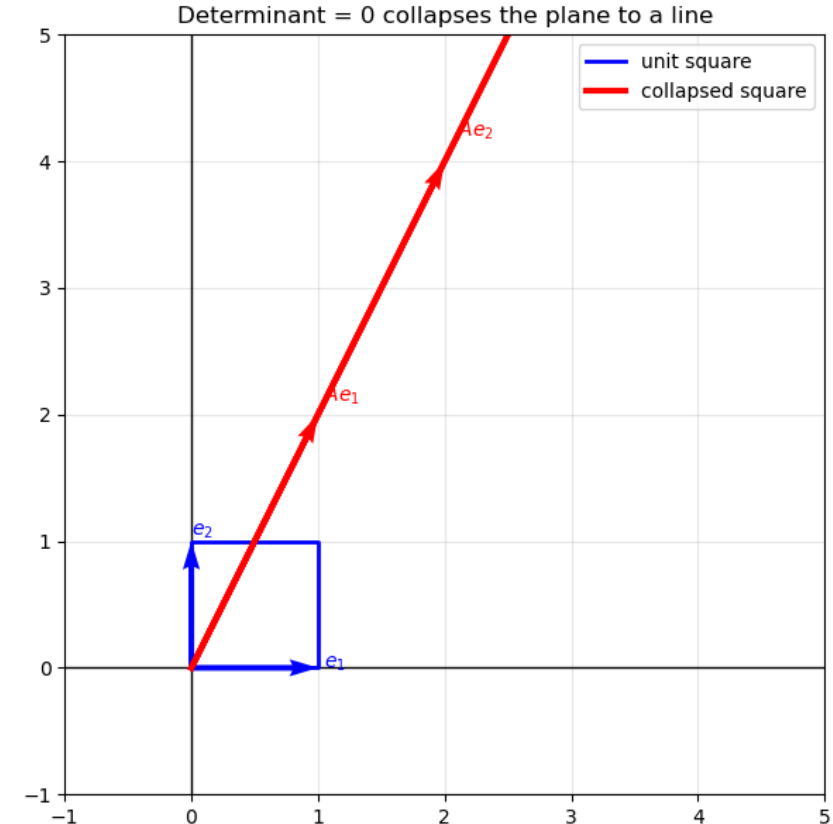
This figure is similar to when I showed how operators transform shapes.

Sometimes we illustrate a transformation by applying it to a few example vectors, such as when we showed rotations. But an operator actually acts on **all vectors in the space**.

Because of this, it is often helpful to think about how the **entire space of points** is transformed.

In this case we look at what happens to the **unit square**. Every vector inside that square is transformed by the operator A . Because the determinant of A is zero, the transformation collapses the square into a **line**.

The reason the line implies A is not invertible is that there isn't a unique mapping from points on a line to a square! So the operator A cannot be undone by feeding in points from the line to the inverse of A , (A^{-1}) the output wouldn't be unique.



HOW TO CALCULATE EIGENVALUES AND EIGENVECTORS

By definition an eigenvector v satisfies

$$Av = \lambda v$$

Rearrange:

$$Av - \lambda v = 0$$

Factor:

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

For a non-zero vector v to exist,

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

HOW TO CALCULATE EIGENVALUES AND EIGENVECTORS

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

By definition an eigenvector v satisfies

$$Av = \lambda v$$

Rearrange:

$$Av - \lambda v = 0$$

Factor:

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

For a non-zero vector v to exist,

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

For example:

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

Eigenvalues

$$\det(A - \lambda I) = 0,$$

$$\begin{aligned} \det(A - \lambda I) &= (4 - 5\lambda + \lambda^2) - 4 = \lambda^2 - 5\lambda = \lambda(\lambda - 5) \det(A - \lambda I) \\ &= (1 - \lambda)(4 - \lambda) - 4A - \lambda I = \begin{pmatrix} 1 - \lambda & 2 \\ 2 & 4 - \lambda \end{pmatrix} \end{aligned}$$

$$\lambda(\lambda - 5) = 0 \Rightarrow \lambda_1 = 0, \lambda_2 = 5$$

(i. e. the equation is zero when λ is 0 or 5)

EIGENVALUES AND VECTORS IN COMPLEX SPACE

Eigenvector for $\lambda = 0$

$$Av = 0, \quad v = (xy)$$

$$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = (x + 2y, 2x + 4y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$x + 2y = 0 \Rightarrow x = -2y$$

Choose $y = 1 \Rightarrow x = -2$:

$$v_0 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$$

HOW TO CALCULATE EIGENVALUES AND VECTORS

Eigenvector for $\lambda = 0$

$$Av = 0, \quad v = (xy)$$

$$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = (x + 2y \quad 2x + 4y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$x + 2y = 0 \Rightarrow x = -2y$$

Choose $y = 1 \Rightarrow x = -2$:

$$v_0 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$$

Eigenvector for $\lambda = 5$

$$\begin{pmatrix} -4 & 2 \\ 2 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = (-4x + 2y \quad 2x - y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$2x - y = 0 \Rightarrow y = 2x$$

Choose $x = 1 \Rightarrow y = 2$:

$$v_5 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

IN CODE

Find eigenvalues:

```
import math # for sqrt

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = math.sqrt(disc)
    λ1 = (tr + root) / 2
    λ2 = (tr - root) / 2
    return λ1, λ2
```

We can write code to do this laborious work for us in a few lines.

Here is a function that finds the eigenvalues for a 2x2 matrix

IN CODE

Find eigenvalues:

```
import math # for sqrt

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = math.sqrt(disc)
    λ1 = (tr + root) / 2
    λ2 = (tr - root) / 2
    return λ1, λ2
```

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

```
def solve_2x2(A, λ):
    a, b = A[0]
    c, d = A[1]
    m11 = a - λ
    m12 = b
    m21 = c
    m22 = d - λ
    if abs(m11) > 1e-12:
        x = -m12 / m11
        y = 1
    else:
        x = -m22 / m21
        y = 1
    return (x, y)
```

A short function to solve
the linear equation

IN CODE

Find eigenvalues:

```
import math # for sqrt

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = math.sqrt(disc)
    λ1 = (tr + root) / 2
    λ2 = (tr - root) / 2
    return λ1, λ2
```

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

```
def solve_2x2(A, λ):
    a, b = A[0]
    c, d = A[1]
    m11 = a - λ
    m12 = b
    m21 = c
    m22 = d - λ
    if abs(m11) > 1e-12:
        x = -m12 / m11
        y = 1
    else:
        x = -m22 / m21
        y = 1
    return (x, y)
```

Simplify the vector

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return abs(a)

def lcd(a, b):
    return abs(a*b)//gcd(a, b)
```

When we solve for eigen vectors it is nice (but not necessary) to show them in their simplest form. So we define the gcd (greatest common divisor function)

IN CODE

Find eigenvalues:

```
import math # for sqrt

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = math.sqrt(disc)
    λ1 = (tr + root) / 2
    λ2 = (tr - root) / 2
    return λ1, λ2
```

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

```
def solve_2x2(A, λ):
    a, b = A[0]
    c, d = A[1]
    m11 = a - λ
    m12 = b
    m21 = c
    m22 = d - λ
    if abs(m11) > 1e-12:
        x = -m12 / m11
        y = 1
    else:
        x = -m22 / m21
        y = 1
    return (x, y)
```

Simplify the vector

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return abs(a)

def lcd(a, b):
    return abs(a*b)//gcd(a, b)
```

Find eigenvector from eigenvalue:

```
def eigenvector_2x2(A, λ):
    x, y = solve_2x2(A, λ)
    x, y = int(round(x)), int(round(y))
    g = gcd(x, y)
    return (x//g, y//g)
```

Now we can find the eigenvector that corresponds to each eigenvalue

IN CODE

Find eigenvalues:

```
import math # for sqrt

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = math.sqrt(disc)
    λ1 = (tr + root)/2
    λ2 = (tr - root)/2
    return λ1, λ2
```

Find eigenvector from

```
def eigenvector_2x2(A, λ):
    x, y = solve_2x2(A - λ*I)
    x, y = int(x), int(y)
    g = gcd(x, y)
    return (x//g, y//g)
```

```
A = [[1, 2], [2, 4]]
```

```
λ1, λ2 = eigenvalues_2x2(A)
```

```
print("Eigenvalues:", str(λ1)+",", λ2)
```

```
v1 = eigenvector_2x2(A, λ1)
```

```
v2 = eigenvector_2x2(A, λ2)
```

```
print("Eigenvector for λ =", str(λ1)+":", v1)
```

```
print("Eigenvector for λ =", str(λ2)+":", v2)
```

Eigenvalues: 5.0, 0.0

Eigenvector for λ = 5.0: (0.5, 1)

Eigenvector for λ = 0.0: (-2.0, 1)

HOW TO CALCULATE EIGENVALUES AND VECTORS IN $\mathbb{C}^2$

Recall our linear operator R that had no real eigenvalues.

$$R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

This rotates vectors by **90°**.

In real space: no vector keeps its direction
therefore **no real eigenvectors exist**

But when we solve

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

we obtain $\lambda = \pm i$

So the eigenvectors live in **complex space**.

HOW TO CALCULATE EIGENVALUES AND VECTORS IN $\mathbb{C}^2$

Rotation in $\mathbb{R}^2$ and complex eigenvalues

Recall our linear operator R that had no real eigenvalues.

$$R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

This rotates vectors by 90° .

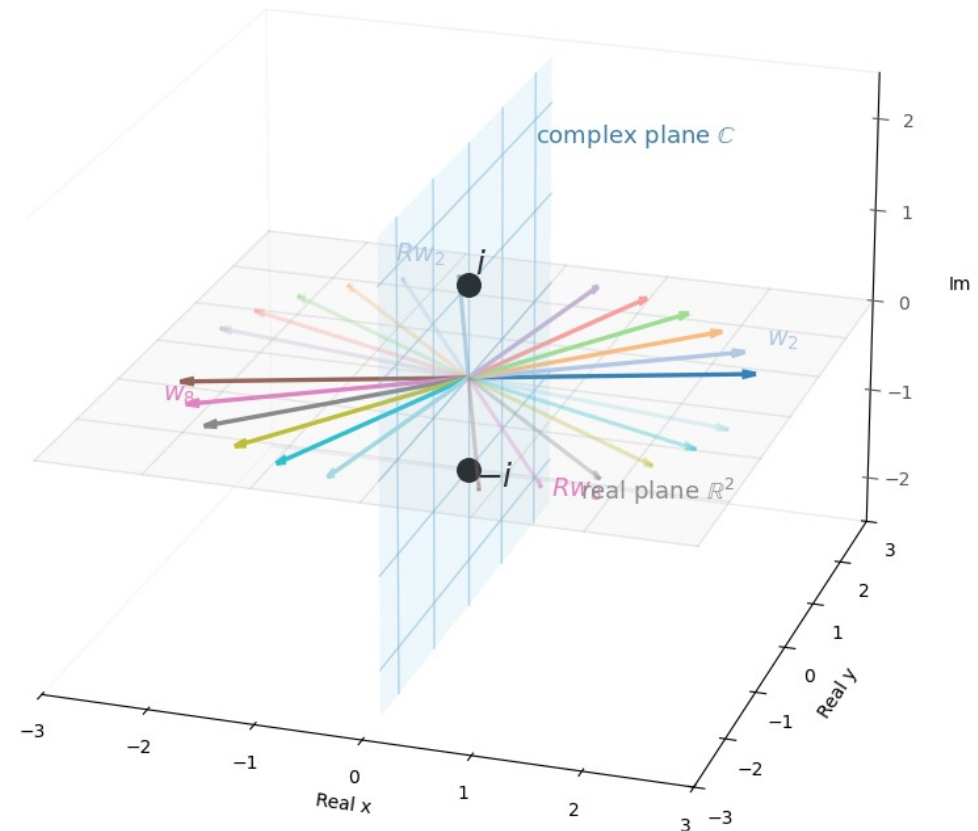
In real space: no vector keeps its direction therefore **no real eigenvectors exist**

But when we solve

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

we obtain $\lambda = \pm i$

So the eigenvectors live in **complex space**.

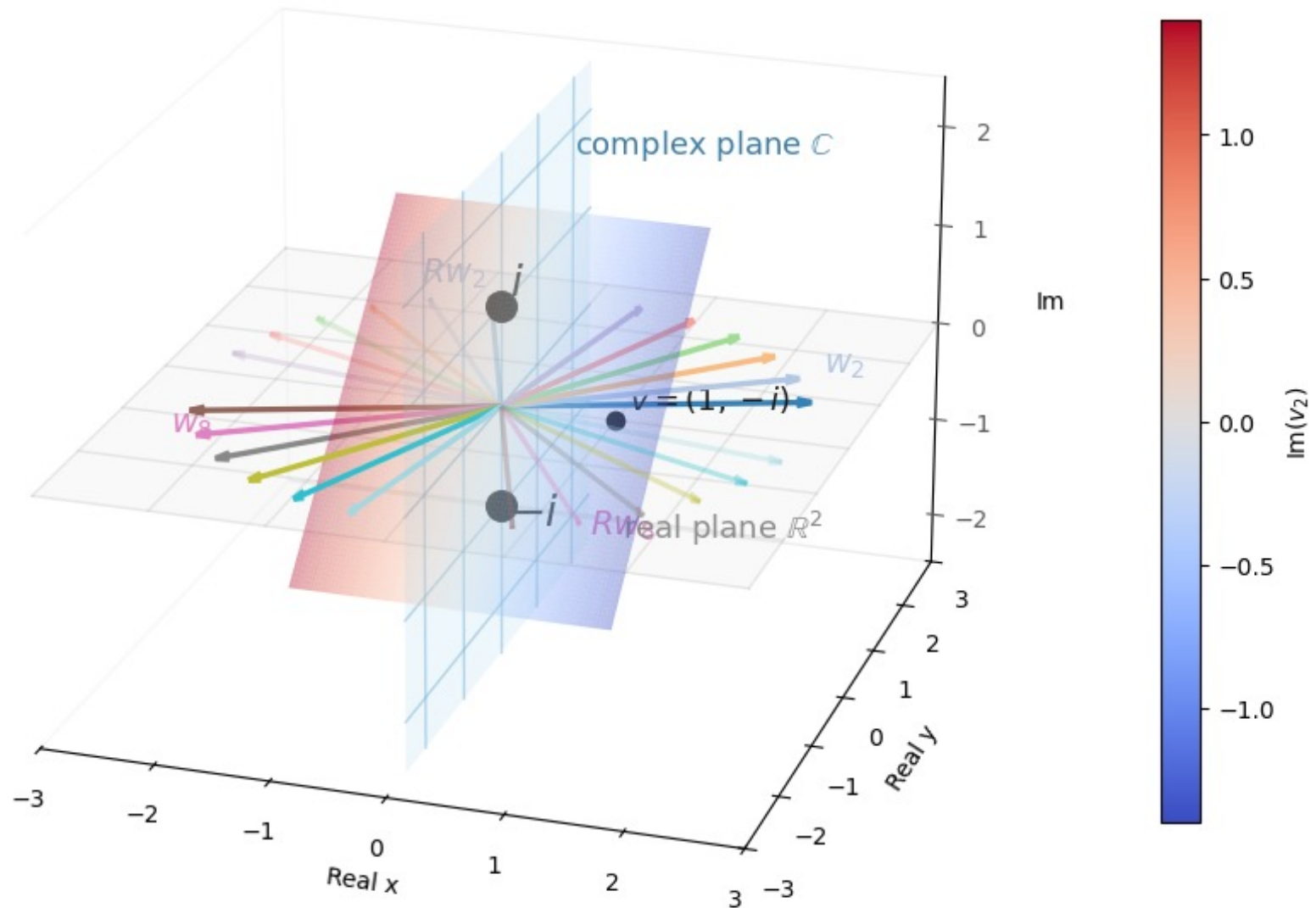


The corresponding eigenvectors are in $\mathbb{R}^4$

HOW TO CALCULATE EIGENVALUES AND VECTORS IN $\mathbb{C}^2$

This figure shows how the rotation matrix rotates every real vector in $\mathbb{R}^4$, so no real eigenvectors exist. The eigenvalues instead appear in the complex plane at i and $-i$.

The coloured surface represents the complex eigenvector line, where colour encodes the 4th dimension needed to visualise vectors in $\mathbb{C}^2$.



Corresponding eigenvectors lie in $\mathbb{R}^4$, shown as a colour-encoded surface.

IN CODE

Find eigenvalues:

```
import cmath # for sqrt

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = cmath.sqrt(disc)
    λ1 = (tr + root) / 2
    λ2 = (tr - root) / 2
    return λ1, λ2
```

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

```
def solve_2x2(A, λ):
    a, b = A[0]
    c, d = A[1]
    m11 = a - λ
    m12 = b
    m21 = c
    m22 = d - λ
    if abs(m11) > 1e-12:
        x = -m12 / m11
        y = 1
    else:
        x = -m22 / m21
        y = 1
    return (x, y)
```

Simplify the vector

```
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return abs(a)

def lcd(a, b):
    return abs(a*b) // gcd(a, b)
```

Find eigenvector from eigenvalue:

```
def eigenvector_2x2(A, λ):
    x, y = solve_2x2(A, λ)
    x, y = int(round(x)), int(round(y))
    g = gcd(x, y)
    return (x//g, y//g)
```

Now we can find the eigenvector that corresponds to each eigenvalue

IN CODE

Find eigenvalues:

```
import cmath # for square roots

def eigenvalues_2x2(A):
    a, b = A[0]
    c, d = A[1]
    tr = a + d
    det = a*d - b*c
    disc = tr*tr - 4*det
    root = cmath.sqrt(disc)
    λ1 = (tr + root)/2
    λ2 = (tr - root)/2
    return λ1, λ2
```

Find eigenvector from

```
def eigenvector_2x2(A, λ):
    x, y = solve_2x2(A - λ*I)
    return (x, y)
```

```
A = [[0, -1],[1, 0]]
```

```
λ1, λ2 = eigenvalues_2x2(A)
print("Eigenvalues:", λ1, ",", λ2)
```

```
v1 = eigenvector_2x2(A, λ1)
v2 = eigenvector_2x2(A, λ2)
print("Eigenvector for λ =", λ1, ":", v1)
print("Eigenvector for λ =", λ2, ":", v2)
```

Eigenvalues: $1j$, $-1j$

Eigenvector for $\lambda = 1j$: $((-0+1j), 1)$

Eigenvector for $\lambda = -1j$: $(-1j, 1)$

corresponds to each complex eigenvalue

) $\rightarrow$ error
1) $\rightarrow 1j$

ation step
a common
variable for i)

envectors are
ectly

t make sense

vector that

Parietal lobe



HERMITIAN OPERATORS

We have seen that...

Real matrices can have **complex eigenvalues**.

Eigenvectors may live in $\mathbb{C}^n$ rather than $\mathbb{R}^n$.

The geometry gets harder to visualise.

Hermitian matrices are the complex analogue of symmetric matrices, and this symmetry forces their eigenvalues to be real, which we can visualise again in $\mathbb{R}^2$ and $\mathbb{R}^3$.

... bringing the parietal lobe back online for 3D processing.

HERMITIAN OPERATORS

Definition

A matrix H is **Hermitian** if

$$H^\dagger = H$$

where $H^\dagger$ is the **conjugate transpose**.

For a matrix

$$H = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Hermitian means

$$H = \begin{pmatrix} a & b \\ \bar{b} & d \end{pmatrix}$$

with $a, d \in \mathbb{R}$ and recall $\bar{b}$ is the conjugate of b .

A matrix H is **Hermitian** if it is **equal to its own adjoint**:

$$H^\dagger = H$$

It's like identity under the adjoint transformation.

Recall: if $z = a + bi$ then its conjugate is $\bar{z} = a - bi$; Adjoint: $A^\dagger = \bar{A}^T$

HERMITIAN OPERATORS



A matrix H is **Hermitian** if it is **equal to its own adjoint**:

$$H^\dagger = H$$

It's like identity under the adjoint transformation.

Charles Hermite (1822–1901)

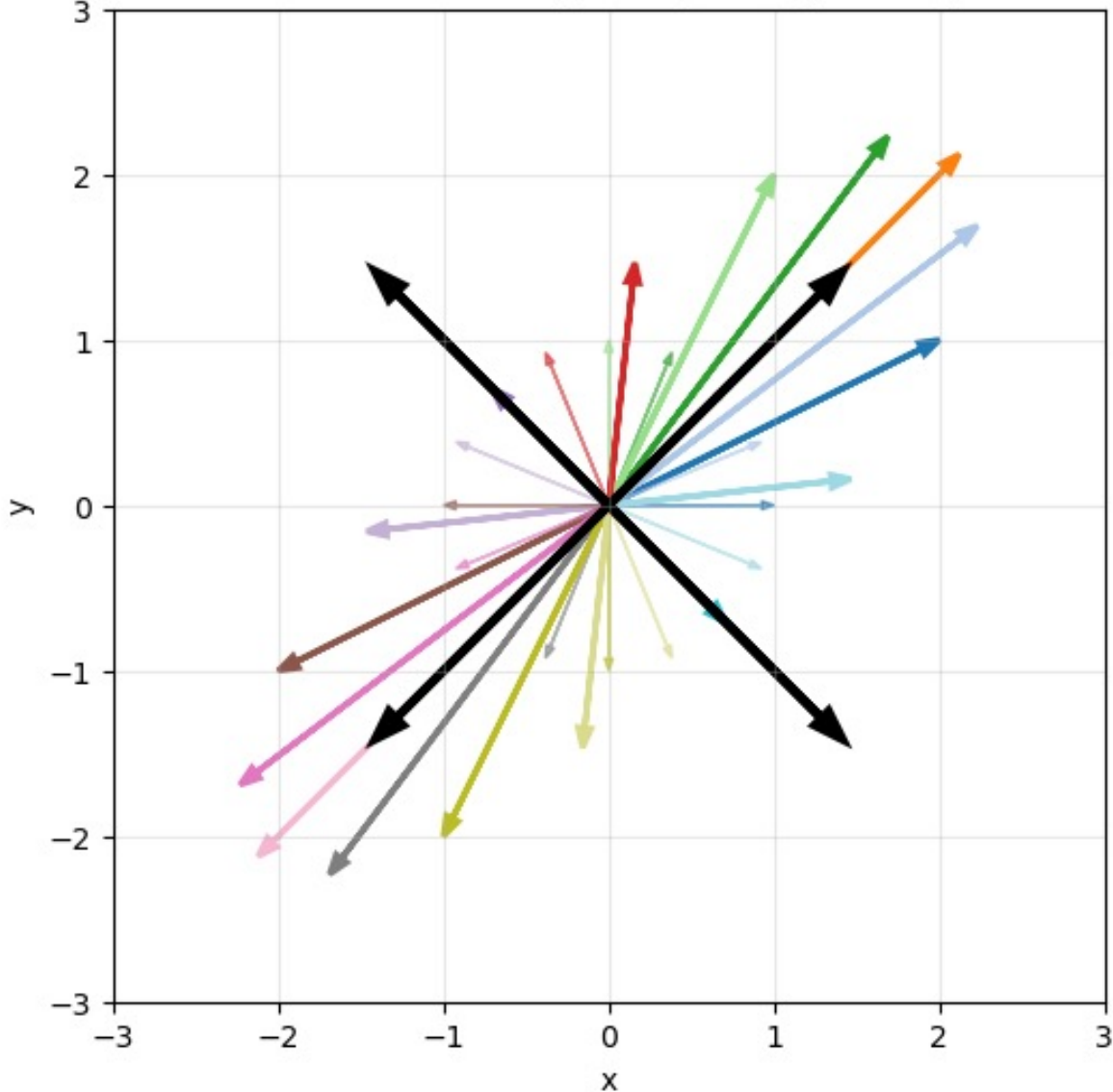
Sur la théorie des formes quadratiques (*On the theory of quadratic forms*), symmetry of complex matrices, 1854
École Polytechnique, Paris

“I turn away with fear and horror from this lamentable plague of functions which have no derivatives.”

in reaction to Karl Weierstrass and the rise of pathological functions in real analysis.

HERMITIAN OPERATORS

Hermitian matrix: stretching along orthogonal eigenvectors



Most matrices do **not** have orthogonal eigenvectors.

Hermitian matrices are special because they always have orthogonal eigenvectors.

The black eigenvectors are orthogonal, so they form an **orthonormal basis**.

Much of mathematics is the search for structures that make difficult ideas easier to reason about.

- **Roman vs positional notation**

Roman numerals: VI + VIII = XIV

Arabic numerals: 6 + 8 = 14 Place value makes arithmetic systematic.

- **Cartesian coordinates**

Geometry → algebra $x^2 + y^2 = r^2$

A circle becomes an equation that can be manipulated algebraically.

- **Leibniz calculus notation** $\frac{dy}{dx}$

Rules such as the chain rule become transparent: $\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$

- **Hermitian operators** $A^\dagger = A$

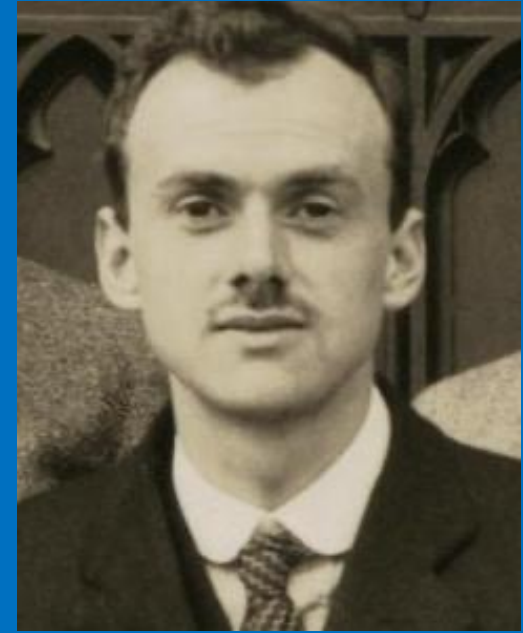
This structure guarantees real eigenvalues and an orthonormal eigenbasis: $A v = \lambda v$

Much of mathematics is the search for structures that make
diff

Paul Dirac

• Re Principles of Quantum
Ror
Ara Mechanics (1930)

• Ca University of Cambridge



y Geo
A c
• Le “The formulation of these laws requires the use of the
Rul mathematics of transformations. **The important things in the
world appear as the invariants** ... of these transformations.”

• H The **eigenvalues** of the Hermitian operators are those **invariants**.

This structure guarantees real eigenvalues and an orthonormal eigenbasis: $A v = \lambda v$

UNITARY OPERATORS

Unitary operators

A linear operator U is **unitary** if

$$U^\dagger U = I$$

Equivalently,

$$\langle Uv, Uw \rangle = \langle v, w \rangle$$

So unitary operators preserve:

- lengths
- angles
- orthogonality

This means

$$\|Uv\| = \|v\|$$

and

$$U^{-1} = U^\dagger$$

Preview: Quantum gates must be unitary, because quantum evolution must preserve the norm of the state vector.

Operators that have these properties preserve the geometry of Hilbert space.

UNITARY OPERATORS

Example. Let's say we want an operator to rotate input vectors by

$$\theta = \frac{\pi}{5}$$

The rotation matrix is:

$$R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

For $\pi/5$:

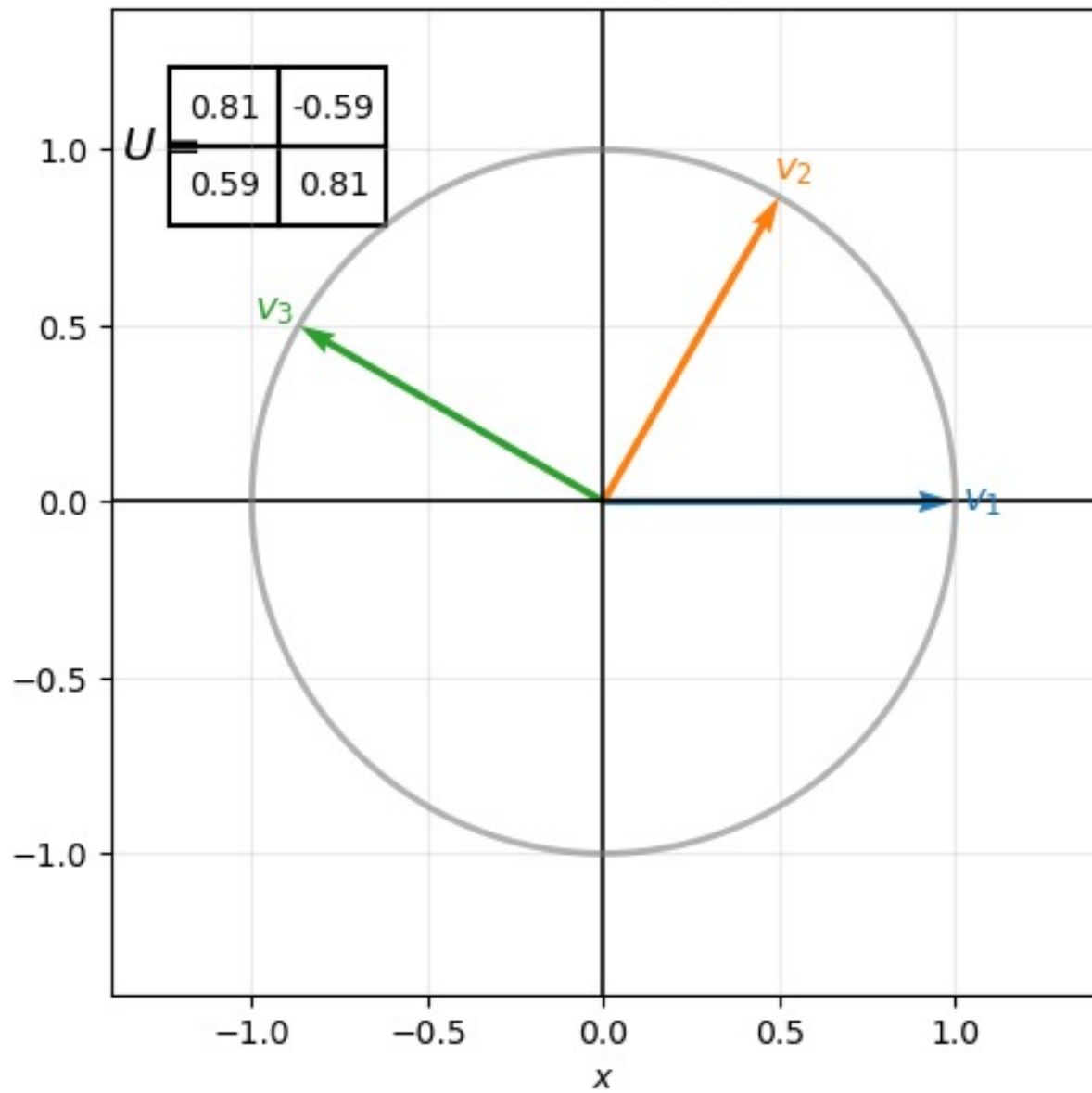
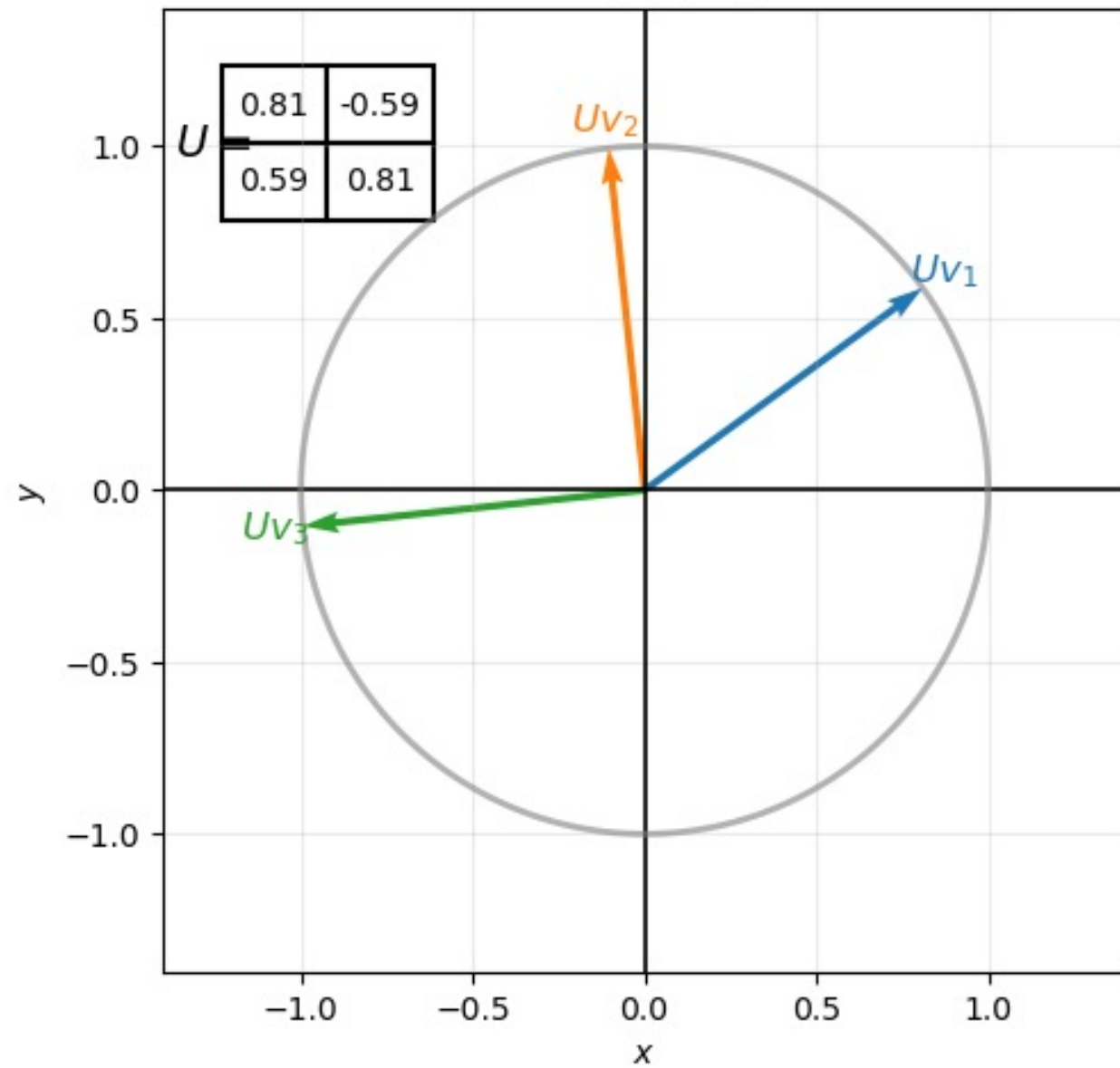
$$R\left(\frac{\pi}{5}\right) = \begin{pmatrix} \cos(\pi/5) & -\sin(\pi/5) \\ \sin(\pi/5) & \cos(\pi/5) \end{pmatrix}$$

Now evaluate the trig functions:

$$\cos(\pi/5) \approx 0.809; \sin(\pi/5) \approx 0.588$$

So the matrix becomes

$$U = \begin{pmatrix} 0.81 & -0.59 \\ 0.59 & 0.81 \end{pmatrix}$$

Before applying U After applying U 

Orthogonal matrices preserve the unit circle in $\mathbb{R}^2$. Unitary operators do the analogous job in complex Hilbert space.

SPECTRAL DECOMPOSITION

The set of eigenvalues of an operator is called its **spectrum**.

So spectral decomposition means

decomposing the operator into its eigenvalue spectrum.

.

SPECTRAL DECOMPOSITION

For a **Hermitian matrix** A :

- all eigenvalues are **real**
- eigenvectors can be chosen **orthonormal**

This means the matrix can be written as

$$A = V\Lambda V^\dagger$$

where

V is a matrix of orthonormal eigenvectors

Λ is a diagonal matrix of eigenvalues.

In explicit matrix form:

$$V = \begin{pmatrix} \vdots & \vdots \\ v_1 & v_2 \\ \vdots & \vdots \end{pmatrix}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

Then

$$A = V\Lambda V^\dagger$$

SPECTRAL DECOMPOSITION

Ordinary diagonalisation (general matrices)

A matrix A is diagonalizable if

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

where

- D is diagonal
- columns of P are eigenvectors.

But this only works if the matrix **has enough independent eigenvectors**.

Many matrices **cannot** be diagonalized this way.

Example

This matrix has **only one eigenvector**, so it is **not diagonalizable**.

Hermitian operators do not have this problem: they always have a full orthonormal set of eigenvectors and therefore can always be diagonalised, which is the basis of the spectral decomposition.

Preview: the eigenvalues will be the outputs of our quantum computations.

SPECTRAL DECOMPOSITION

```
# Example: Matrix that can't be diagonalised  
#-----
```

```
A = [[1, 1],  
     [0, 1]]
```

```
# Compute eigenvalues
```

```
 $\lambda_1, \lambda_2 = \text{complex\_eigenvalues\_2x2}(A)$ 
```

```
print("Eigenvalues:",  $\lambda_1$ , ",",  $\lambda_2$ )
```

```
# Compute eigenvectors
```

```
 $v_1 = \text{complex\_eigenvector\_2x2}(A, \lambda_1)$ 
```

```
 $v_2 = \text{complex\_eigenvector\_2x2}(A, \lambda_2)$ 
```

```
print("Eigenvector for  $\lambda =$ ",  $\lambda_1$ , ":",  $v_1$ )
```

```
print("Eigenvector for  $\lambda =$ ",
```

Eigenvalues: $(1+0j)$, $(1+0j)$

ZeroDivisionError

Since the matrix has just one repeated eigenvalue we can't write a matrix with two eigenvalues on the diagonal.

SPECTRAL DECOMPOSITION

Key Identity:

$$A = V\Lambda V^\dagger$$

or equivalently

$$V^\dagger AV = \Lambda$$

The operator is a diagonal matrix in the orthonormal basis.

I.e. the things that matter are all on the diagonal line of the matrix.

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

THE OUTER PRODUCT

Definition

For vectors v and w

$$vw^\dagger$$

produces a **matrix**.

Example

$$v = (v_1 v_2), \quad w^\dagger = \begin{pmatrix} \overline{w_1} \\ \overline{w_2} \end{pmatrix}$$

$$vw^\dagger = \begin{pmatrix} v_1 \overline{w_1} & v_1 \overline{w_2} \\ v_2 \overline{w_1} & v_2 \overline{w_2} \end{pmatrix}$$

The overbar denotes the complex conjugate. The adjoint $v^\dagger$ means transpose composed with complex conjugation.

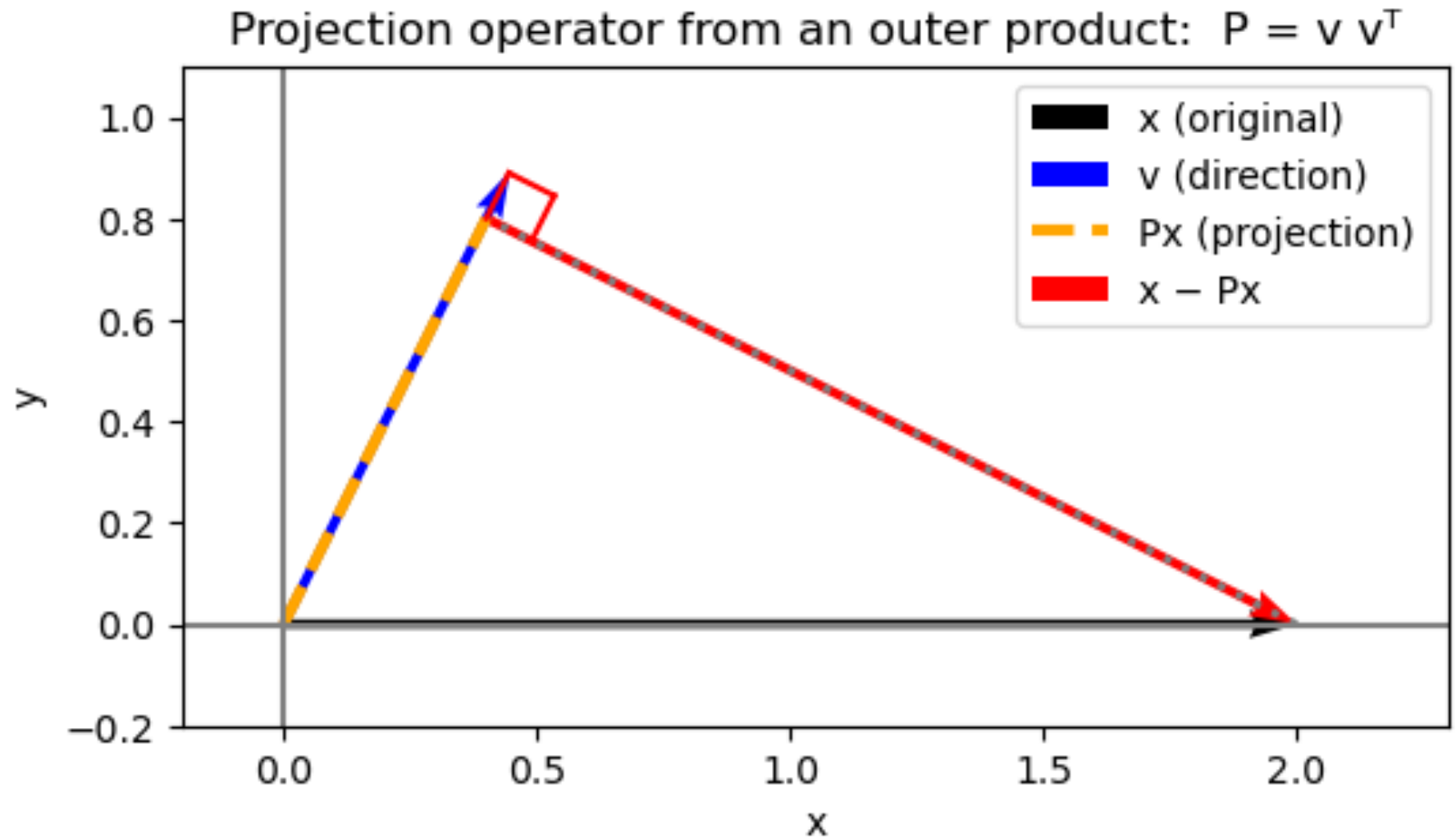
- Inner product $\rightarrow$ number
- Outer product $\rightarrow$ matrix

The outer product gives us the component of a vector in a particular direction.

THE OUTER PRODUCT

We are asking:
“How much of
vector x points in
that direction?”

The projection is
the part that
points that way.

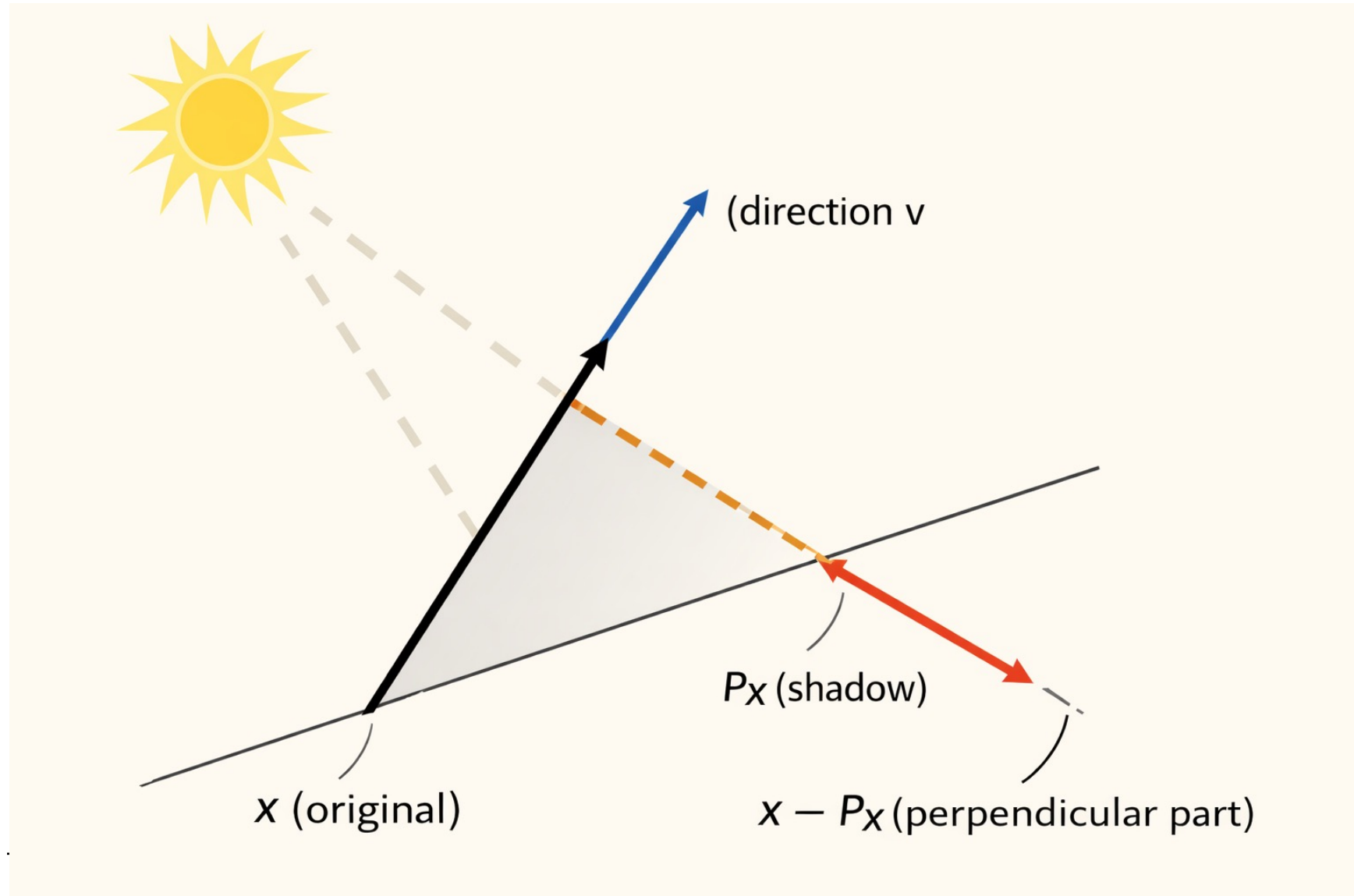


THE OUTER PRODUCT

The original vector $\mathbf{x}$ (black) can be decomposed relative to the direction $\mathbf{v}$ (blue). When light shines from the sun, $\mathbf{x}$ casts a shadow onto the line defined by $\mathbf{v}$.

This shadow is the **projection** $P \mathbf{x}$ (orange dashed segment ending in the red arrow). The remaining component $\mathbf{x} - P \mathbf{x}$ (red) is the part of the vector perpendicular to that direction.

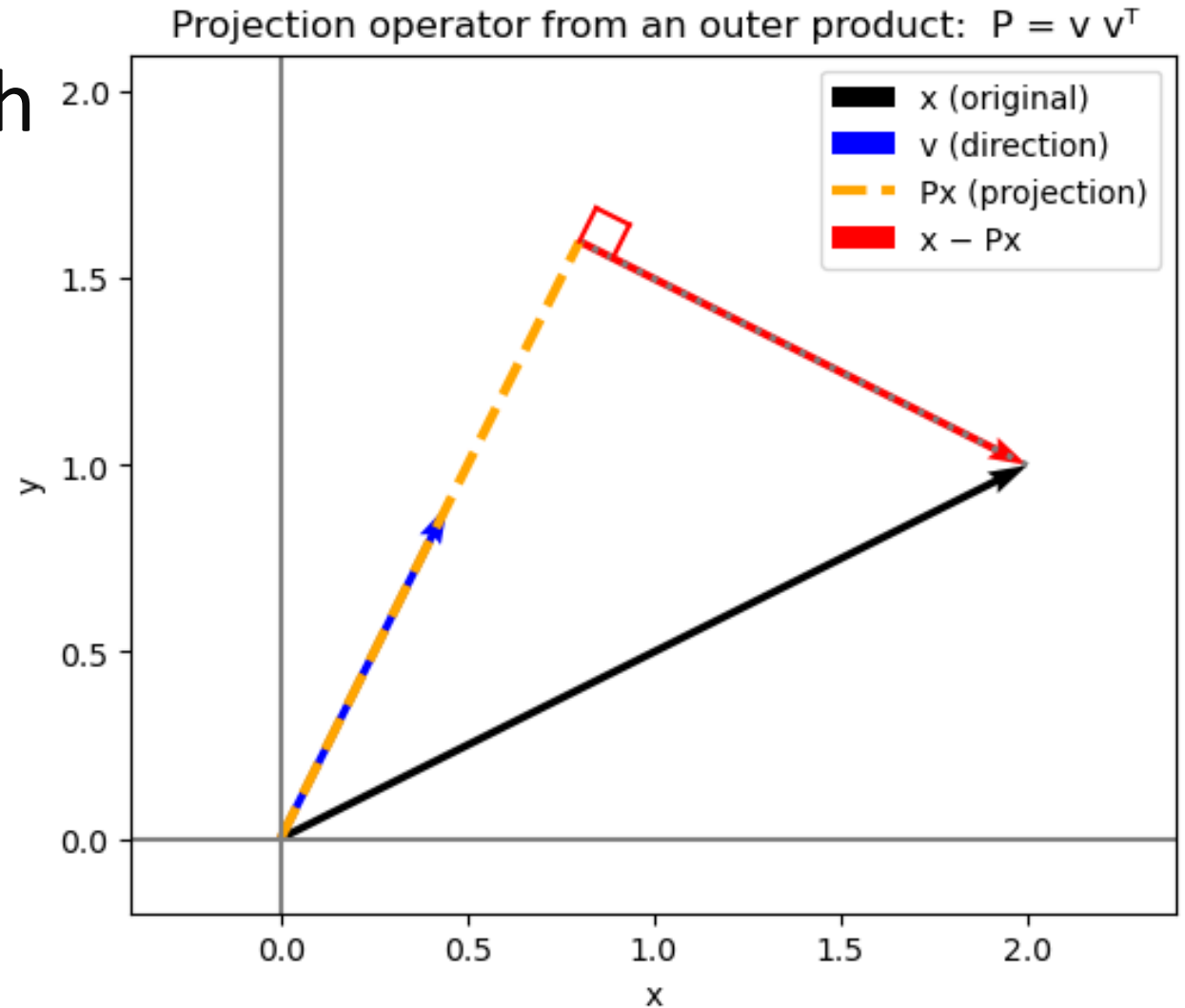
The projection operator keeps only the component of a vector that lies along the chosen direction.



THE OUTER PRODUCT

We are asking: “How much of vector x points in that direction?”

The projection is the part that points that way.



SPECTRAL DECOMPOSITION

For a Hermitian operator

$$A = \sum_i \lambda_i v_i v_i^\dagger$$

Key properties

- Eigenvectors are orthonormal
- They form a complete basis
- Any vector can be decomposed in this basis

Action of the operator

$$Ax = \sum_i \lambda_i c_i v_i$$

The operator scales each eigenvector component by its eigenvalue.

TENSOR PRODUCTS

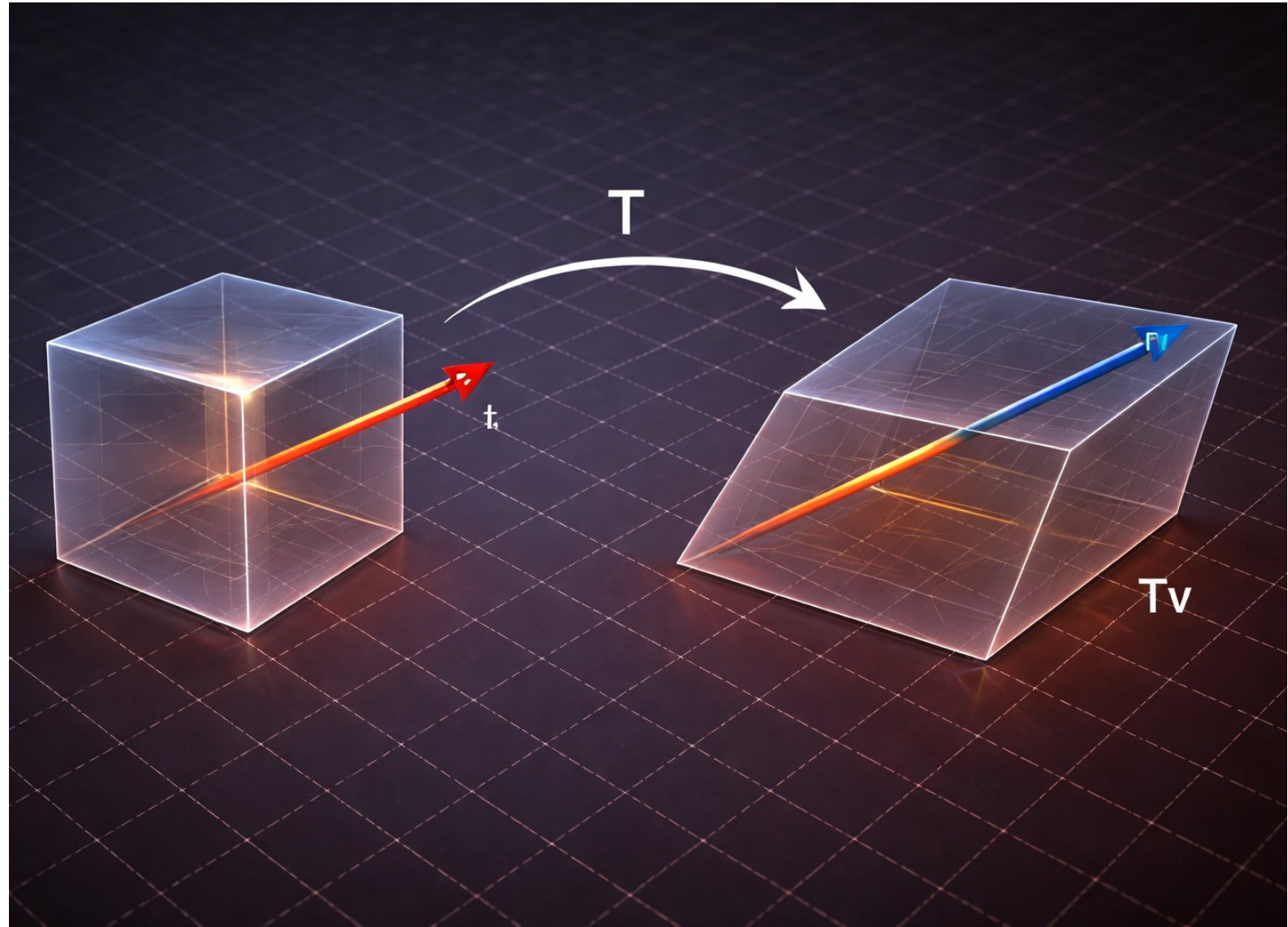
- Recall from earlier that vectors are a mathematical object with direction and length.
 - We can represent a vector as a sequence of values, e.g, $\vec{v} = \begin{pmatrix} 3 \\ 8 \end{pmatrix}$ but the values depend on the coordinate system we chose.
 - If we rotate or change the coordinate axes, the numbers change, even though the vector itself is the same geometric object.
 - The relationship between tensors and matrices is similar.
 - A tensor is the underlying mathematical object that represents the linear interaction of vectors.
 - A matrix is a multi-dimensional array of numbers that represents that tensor once we choose a coordinate system.
 - So just as coordinates represent vectors, matrices represent tensors.
-

TENSORS

A **rank-2 tensor** acts as a linear transformation of vectors, changing their direction and magnitude.

When we choose a coordinate system, this transformation is represented by a **matrix**.

Conceptual art showing 2D tensor distorting vector space.



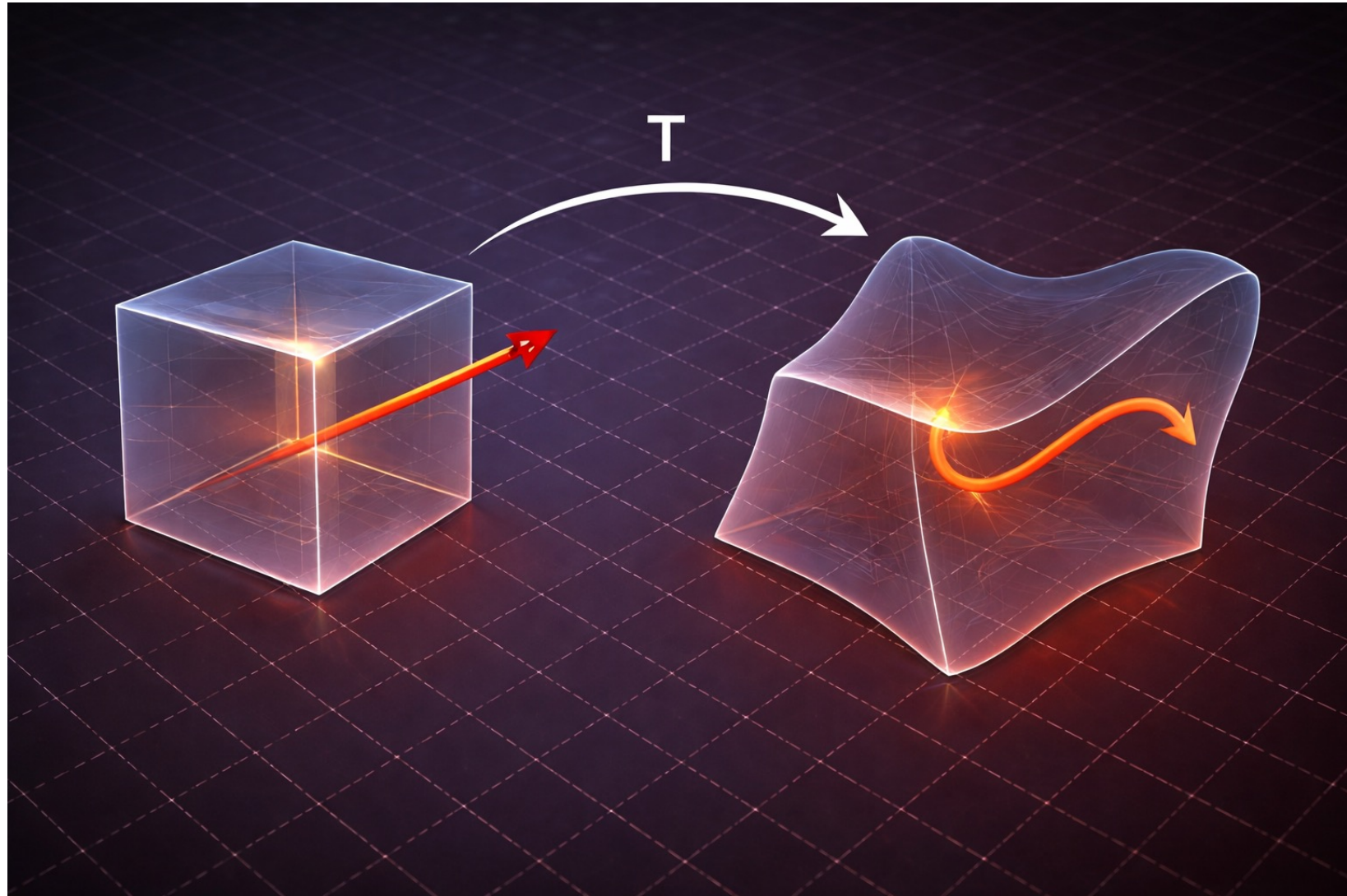
NOT A TENSOR

This concept art shows a **nonlinear transformation**.

Straight lines bend, so the mapping is **not linear** and cannot be represented by a **tensor (matrix)**.

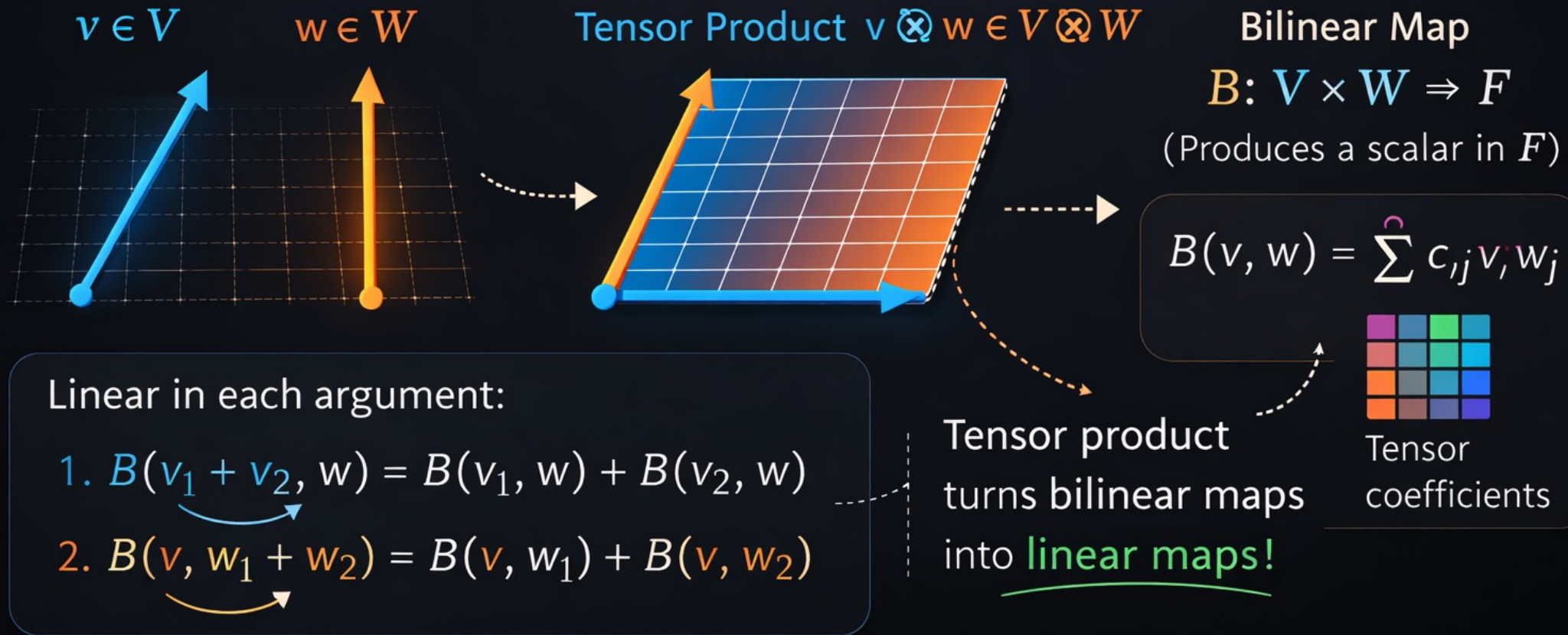
(However, nonlinear transformations can often be approximated locally by linear maps.

In calculus, we approximate a curved transformation with many infinitesimal linear pieces. This is the idea behind derivatives and integrals. We won't care about that in this course.)



A **Bilinear Map**: A Function of Two Vectors That Is **Linear** in Each Argument

Scalar field F = the set of numbers used to scale vectors
(**Real numbers** $\mathbb{R}$ or Complex numbers $\mathbb{C}$ - in quantum mechanics we use $\mathbb{C}$)



TENSOR PRODUCTS

Let V and W be vector spaces over a scalar field $\mathbb{F}$.

The tensor product space
 $V \otimes W$

is a vector space together with a map

$$\otimes: V \times W \rightarrow V \otimes W$$

such that

$$(v, w) \mapsto v \otimes w$$

and the map is bilinear.

The tensor product of matrices is **not** ordinary matrix multiplication.

The tensor product **combines transformations acting on different spaces.**

The product operator takes operators as its arguments.

TENSOR PRODUCTS ARE HIGHER-ORDER FUNCTIONS

The tensor product can combine **vector spaces** and **operators**.

Combining vector spaces

If V, W are vector spaces then the tensor product constructs a new space

$$V \otimes W$$

whose dimension satisfies

$$\dim(V \otimes W) = \dim(V) \dim(W)$$

Combining operators

If $A : V \rightarrow V, B : W \rightarrow W$

then the tensor product produces a new operator

$$A \otimes B : V \otimes W \rightarrow V \otimes W$$

defined by

$$(A \otimes B)(v \otimes w) = (Av) \otimes (Bw)$$

PROGRAMMING ANALOGY

In computer science we often write **higher-order functions**:

```
def pair_map(f, g):  
    def combined(x, y):  
        return (f(x), g(y))  
    return combined # return a function  
  
def square(x):  
    return x * x  
  
def double(x):  
    return 2 * x  
  
h = pair_map(square, double)  
print(h(3.0, 4.0))
```

(9.0, 8.0)

The tensor product constructs a new operator

defined by

$$(A \otimes B)(v \otimes w) = (Av) \otimes (Bw)$$

TENSOR PRODUCTS ARE HIGHER-ORDER FUNCTIONS

Example:

Let

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix}$$

Dimension Rule

$$A \in \mathbb{R}^{2 \times 2}, \quad B \in \mathbb{R}^{2 \times 2}$$

$$A \otimes B \in \mathbb{R}^{4 \times 4}$$

Both represent **linear operators on** $\mathbb{R}^2$

The tensor product operator

$$A \otimes B$$

is defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix} = \begin{pmatrix} 1B & 2B \\ 3B & 4B \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 5 & 5 \\ 6 & 7 \end{pmatrix} & 2 \begin{pmatrix} 5 & 5 \\ 6 & 7 \end{pmatrix} \\ 3 \begin{pmatrix} 5 & 5 \\ 6 & 7 \end{pmatrix} & 4 \begin{pmatrix} 5 & 5 \\ 6 & 7 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{pmatrix}$$

WE ARE READY TO LEARN ABOUT QUANTUM COMPUTING!

A single qubit lives in the Hilbert space

$$\mathbb{C}^2$$

with basis

$$|0\rangle = (10) \quad |1\rangle = (01)$$

A general state is

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Two Qubits

If we combine two qubits, the state space must represent all combinations of their amplitudes.

The correct construction is the tensor product space

$$\mathbb{C}^2 \otimes \mathbb{C}^2$$

which has dimension

$$2 \times 2 = 4$$
