Linear Algebra

Vectors, vector spaces, matrices, coordinates, and transformations

Yordan Darakchiev

Technical Trainer

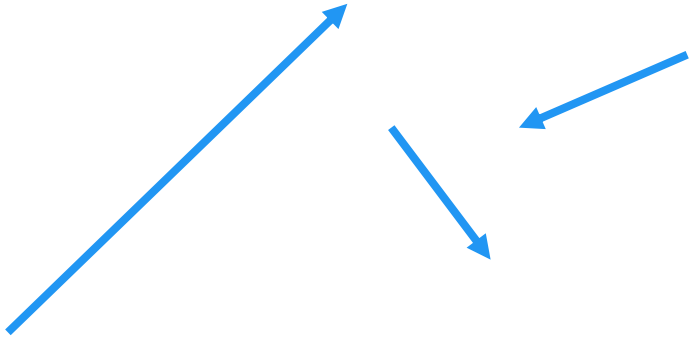
iordan93@gmail.com

Table of Contents

Vectors

Concrete and abstract

Vector Definitions

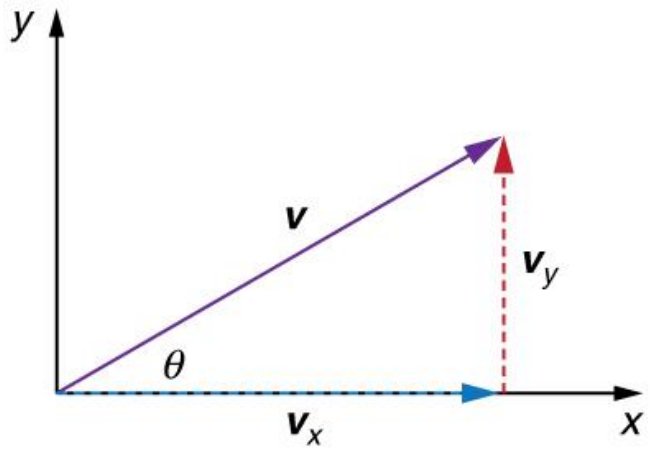


- "Physics definition"
  - A pointed segment in space
- "Computer science definition"
  - A list of objects (usually numbers)
  - Dimensions = length

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad \begin{bmatrix} -3, 8 \\ 0 \\ 5 \end{bmatrix}$$

- Math definition
  - Encompasses both, and allows even more abstraction: $\vec{v}$
  - Vectors can be added and multiplied
    - By numbers and other vectors
  - Similar to how we defined a field
- Another perspective
  - Transformations
  - Actually, things a just a little more complicated…
    - You can look up "tensors" if you're interested
    - We'll talk a little about tensors later

Vector Components



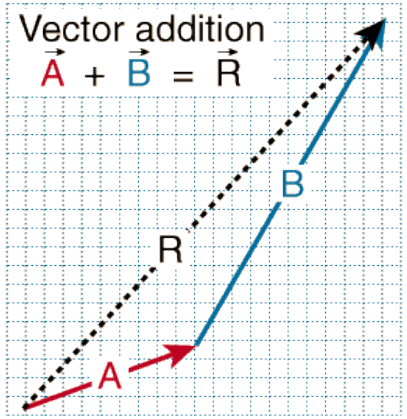
- The distances to all coordinate axes: v_x, v_y
- Equivalent to $\begin{bmatrix} v_x \\ v_y \end{bmatrix}$
  - Polar coordinates: $v = |\vec{v}|, \theta$
- Finding components
  - Pythagoras

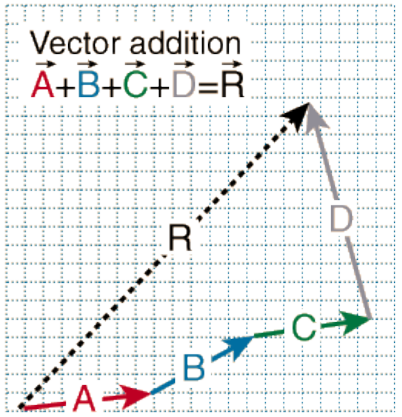
    $v_x = v\cos(\theta),\ v_y = v\sin(\theta)$
- Finding the polar form (magnitude, direction)

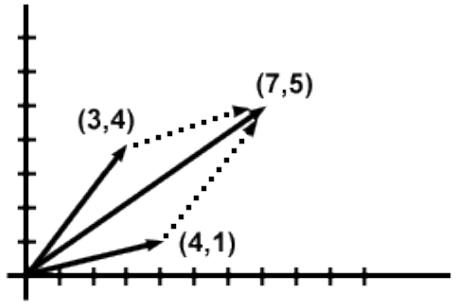
  $v = \sqrt{v_x^2 + v_y^2},\ \theta = \tan^{-1}\left(\frac{v_y}{v_x}\right)$
- All these operations generalize to more than 2 dimensions
- **Note:** We usually denote vectors by $\vec{v}$ or with bold type: $\boldsymbol{v}$
  - Another notation: Latin letters for vectors, Greek letters for numbers
  - Reason: The vector $\boldsymbol{v}$ and its length v can be easily confused

Vector Operations

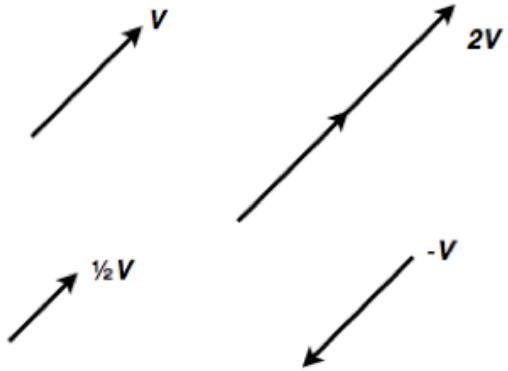
- Addition
  - Result:
    length = distance from start to end,
    direction: start → end
  - In component form
    - Sum all components for every direction



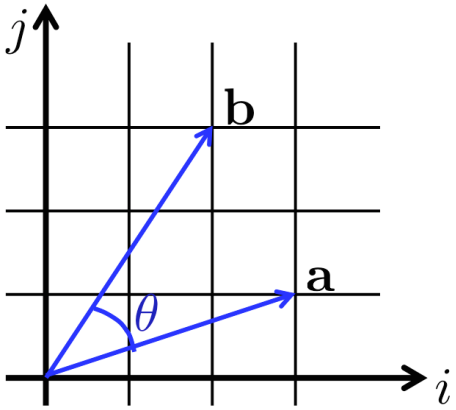




- Multiplication by a number (scalar)
  - Result:
    length = scaled length,
    direction: same (if scalar ≥ 0), opposite otherwise
  - In component form
    - Multiply each component by the number



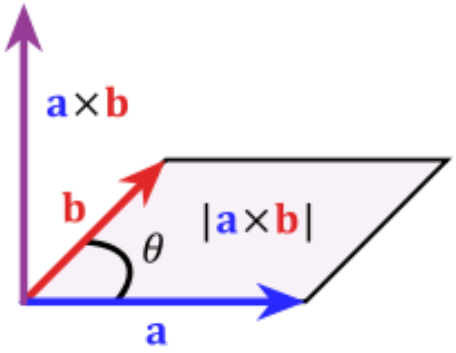
Vector Operations (2)



- Scalar product of two vectors
  - Also called dot product or inner product
  - Result: scalar
  - Definition: $\vec{a}.\vec{b} = |\vec{a}||\vec{b}|\cos(\theta)$
  - Using the vector components:

$$\vec{a}.\vec{b} = \sum_{i=1}^{n} a_i b_i$$

- Vector product of two vectors
  - Also called cross product
  - Result: vector, perpendicular to both initial vectors
  - Definition $\vec{a} \times \vec{b} = |\vec{a}||\vec{b}|\sin(\theta)\vec{n}$
    - $\vec{n}$ – normal vector
    - Magnitude: $|\vec{a}||\vec{b}|\sin(\theta)$ = volume of parallelogram between $\vec{a}$ and $\vec{b}$
    - Direction: coincides with the direction of $\vec{n}$



Vector Spaces

Finding yourself in space

Vector Space

- A field (usually $\mathbb{R}$ or $\mathbb{C}$): F
- A set of elements (vectors): V
- Operations
  - Addition of two vectors: $w = u + v$
  - Multiplication by an element of the field: $w = \lambda u$
- A “checklist” of eight axioms
- We read this as "vector space (or linear space) V over the field F"

Examples of vector spaces

- Coordinate space, e.g. real coordinate space $\mathbb{R}^n$
  - n-dimensional vectors
- Infinite coordinate space $\mathbb{R}^\infty$
  - Vectors with infinitely many components
- Polynomial space
  - All polynomials of variable x with real coefficients $\mathbb{R}[x]$
- Function space

Linear Combinations

- Vectors $v_1, v_2, \ldots, v_n$
- Numbers (scalars) $\lambda_1, \lambda_2, \ldots, \lambda_n$
- **Linear combination**
  - The sum of each vector multiplied by a scalar coefficient

    $$\lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_n v_n = \sum \lambda_i v_i$$

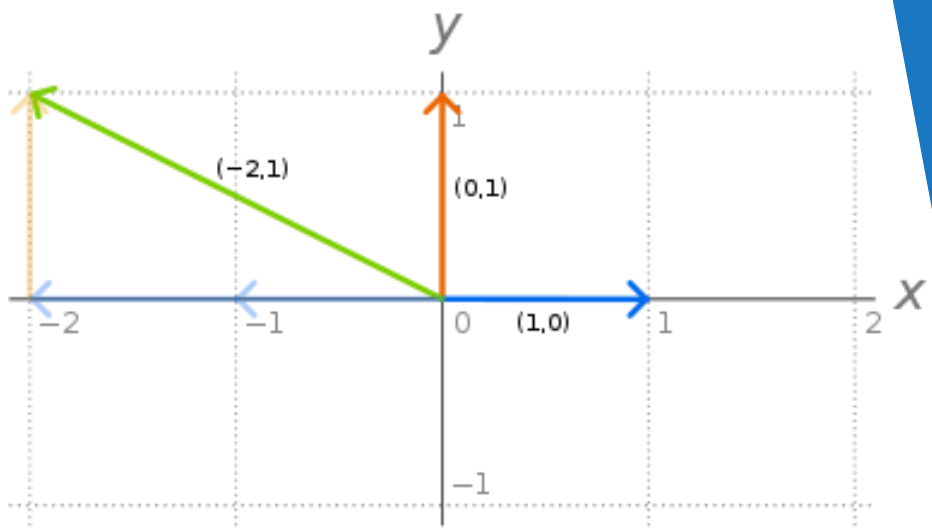
  - Why linear? No fancy functions, no vector multiplications
- **Span** (linear hull) of vectors: the set of all their linear combinations
- Linear (in)dependence
  - The vectors $v_1, \ldots, v_n$ are **linearly independent** if the only solution to the equation $\lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_n v_n = \vec{0}$ is $\lambda_1 = 0, \lambda_2 = 0, \ldots, \lambda_n = 0$
  - Conversely, they are **linearly dependent** if there is a non-trivial linear combination which is equal to zero
- Example

  $u = (2, -1, 1),\ v = (3, -4, -2),\ w = (5, -10, -8)$

  $w = -2u + 3v \Rightarrow 2u - 3v + 1w = 0$

Basis Vectors

- Consider $\hat{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \hat{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



- Now consider the vector $a = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$
- We can see that we can express a as the linear combination $a = -2\hat{i} + 1\hat{j}$

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

- **Basis vectors**
  - Linearly independent
  - Every other vector in the space can be represented as their linear combination
    - This linear combination is **unique**
- **Each vector space has a basis**
- Each pair of two LI vectors forms a basis in 2D coordinate space
  - More generally, each set of n LI vectors forms a basis in n-dimensional vector space

Matrices

Which pill are you going to take?

Matrix

- A rectangular table of numbers
- Dimensions: rows × columns
  - May also be infinite
- Examples:

$$A = \begin{bmatrix} 1 & 3 & 5 \\ -2 & 4.2 & 8 \end{bmatrix} \quad B = \begin{bmatrix} 2 & -1 & 1 \\ 4 & 7 & 12 \\ 0 & 5 & -3 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 2 & 4 & 3 \end{bmatrix} \quad C = \begin{bmatrix} 3 \\ 2 \\ -5 \end{bmatrix}$$

  - R – row vector, C – column vector
- Elements $A = \{a_{ij}\}$
- Some thoughts about dimensions
  - Scalars have no dimensions: 2; 3; 18; -42; 0,5
  - Vectors have one dimension: $v = \{v_i\}$
  - Matrices have two dimensions: $A = \{a_{ij}\}$
  - A generalization of this pattern to many dimensions is called a **tensor**
    - Tensors are quite more complicated than this, but for almost all purposes it's OK to think about them as multidimensional matrices

Matrix Operations

- Addition (the dimensions must be the same)

$$A = \begin{bmatrix} 2 & 3 & 7 \\ 8 & 9 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & -3 & 0 \\ 2 & -4 & 1 \end{bmatrix} \Rightarrow A+B = \begin{bmatrix} 2+1 & 3-3 & 7+0 \\ 8+2 & 9-4 & 1+1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 7 \\ 10 & 5 & 2 \end{bmatrix}$$

- Multiplication by a scalar

$$\lambda = 2, A = \begin{bmatrix} 2 & 3 & 7 \\ 8 & 9 & 1 \end{bmatrix} \Rightarrow \lambda A = \begin{bmatrix} 2.2 & 2.3 & 2.7 \\ 2.8 & 2.9 & 2.1 \end{bmatrix} = \begin{bmatrix} 4 & 6 & 14 \\ 16 & 18 & 2 \end{bmatrix}$$

- All $m \times n$ matrices form a vector space
- Transposition: turning rows into columns and vice versa
  - The transpose of a matrix is denoted by an upper index T

$$A^T = (a_{ij})^T_{m \times n} = (a_{ji})_{n \times m}$$

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

Matrix Multiplication

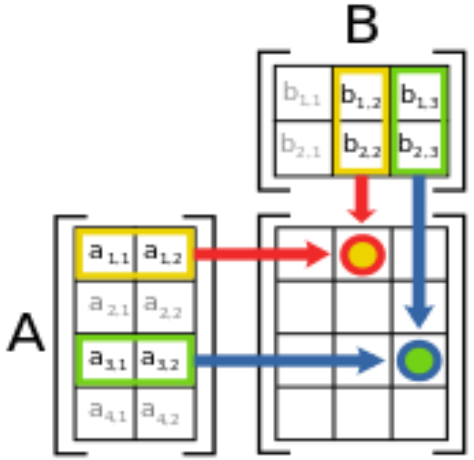
- The dimensions must match: $A_{m\times p}B_{p\times n} = C_{m\times n}$
  - "Inner dimensions" are the same; result has "outer dimensions"
- Definition: $c_{ij} = \sum_{k=1}^{p} a_{ik}b_{kj}$
- Example

$$A = \begin{bmatrix} 2 & 3 & 7 \\ 8 & 9 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 2 & 0 & 1 \\ -3 & -4 & 1 & 3 \\ 2 & 0 & 1 & 1 \end{bmatrix}$$

$$AB = \begin{bmatrix} 2.1 + 3.(-3) + 7.2 & 2.2 + 3.(-4) + 7.0 & 2.0 + 3.1 + 7.1 & 2.1 + 3.3 + 7.1 \\ 8.1 + 9.(-3) + 1.2 & 8.2 + 9.(-4) + 1.0 & 8.0 + 9.1 + 1.1 & 8.1 + 9.3 + 1.1 \end{bmatrix}$$

$$AB = \begin{bmatrix} 7 & -8 & 10 & 18 \\ -17 & -20 & 10 & 36 \end{bmatrix}$$

- Note that $AB \neq BA$
  - In this case, we can't even multiply BA
  - We say that matrix multiplication is not commutative
    - Compare with multiplication of numbers: 5.3 = 3.5 → commutative



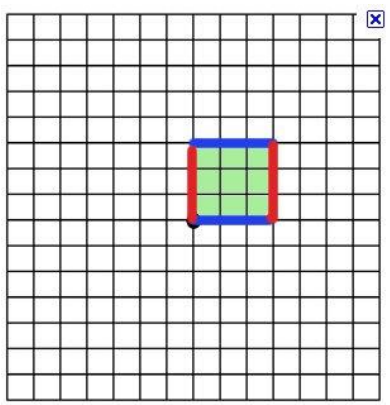
Matrix Operations in numpy

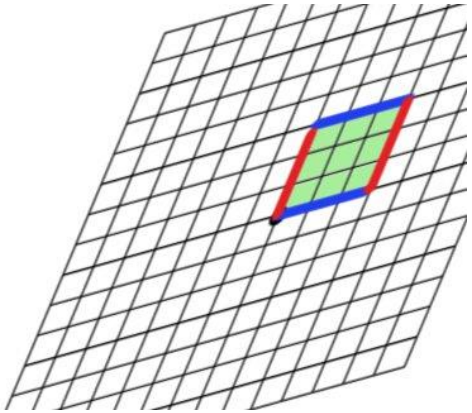
- In numpy we can use `dot()` for matrix multiplication and dot products
- Note: Whenever possible, use numpy arrays instead of Python lists
  - They perform much faster
  - They make the code a lot cleaner

```
A = np.array([
  [2, 3, 7],
  [8, 9, 1]
])
B = np.array([
  [1, -3, 0],
  [2, -4, 1]
])
print(A + B)
print(2 * A)
print(A * B) # Element-wise multiplication
print(A.dot(B)) # Error: incompatible dimensions
print(A.dot(B.T)) # Matrix multiplication
```

Linear Transformations

- **Transformation**
  - A mapping (function) between two vector spaces $V \rightarrow W$
  - Special case: mapping a space onto itself: $V \rightarrow V$
    - This is called a linear operator
  - Each vector of V gets mapped to a vector in W
    - Intuition: think about vectors as points



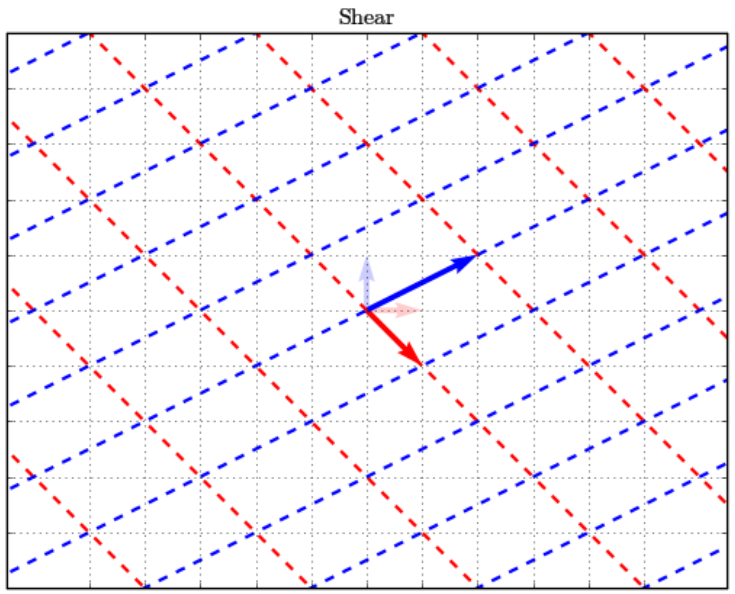


- **Linear**
  - Only linear combinations are allowed
  - The origin remains fixed
  - All lines remain lines (not curves)
  - All lines remain evenly spaced (equidistant)
- Each space has a basis
  - All other vectors can be expressed as linear combinations of the basis vectors
  - **If we know how basis vectors are transformed, we can transform every other vector**

Linear Transformations (2)

- Consider the transformation

$$\hat{i}' = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \hat{j}' = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



- Consider another vector
  - Old basis:

$$v = v_x\hat{i} + v_y\hat{j}$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = v_x \begin{bmatrix} 1 \\ 0 \end{bmatrix} + v_y \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

  - New basis:
    - **Same coefficients, new basis vectors**

$$v' = v_x\hat{i}' + v_y\hat{j}'$$

$$\begin{bmatrix} v'_x \\ v'_y \end{bmatrix} = v_x \begin{bmatrix} 1 \\ -1 \end{bmatrix} + v_y \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

- This operation is called applying the **linear transformation**
  - Note how applying a linear transformation and change of basis are the same

Transformation Matrices

- Consider the same transformation

$$\hat{i}' = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \hat{j}' = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

- We applied the linear transformation by taking dot products
  - Therefore, we can describe it in another way – using a matrix
  - This is called the matrix of the linear transformation
  - Its **columns** denote where the **basis vectors** go

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

- Now, applying the transformation to a vector is the same as **multiplying the matrix times the original vector** (in that order!)

$$v' = Tv$$

- Example:

$$T = \begin{bmatrix} 3 & 2 \\ -2 & 1 \end{bmatrix}, \; v = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

$$v' = \begin{bmatrix} 3 & 2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 29 \\ -3 \end{bmatrix}$$

Multiple Transformations

- We can apply many transformations, one right after the other
  - Result: composite transformation
  - We do this by multiplying **on the left** by the matrix of each transformation
  - So, matrix multiplication ≡ applying many transformations
- To visualize transformations, you can use the code in the **`visualize_transformation.py`** file
- Intuition
  - Apply each transformation in order
  - After the last one, record where the basis vectors land
  - The new matrix is the matrix of the composite transformation
- We can either apply all transformations one by one or just the resulting transformation
  - Note that the second is much, much faster; and gives the same result
- This is especially useful in computer graphics

Multiple Transformations (2)

- **Example:** rotation, then shearing

$$R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \; S = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

- Apply rotation to a vector

$$v' = Rv = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

- Apply shear to the resulting vector

$$v'' = Sv' = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} v'_x \\ v'_y \end{bmatrix}$$

- This is the same as

$$v'' = SRv = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

- The new transformation matrix is

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

Determinant

- A linear transformation stretches space
  - Since lines remain parallel and equidistant, space is stretched equally, no matter where we look
  - A characteristic of the space is “the unit area” – a 1x1 square
  - The transformation will change this area
- **Determinant** – a measure of how much the unit area changes
  - Scalar value
  - Defined only for square matrices
  - For more than two dimensions, area → volume
- The determinant of a matrix A is denoted $\det(A)$
  - Written with straight lines
  - There are “easy” computational rules for 2D and 3D matrices
  - Generally, the determinant is given by a formula
- The determinant has very useful properties
  - Especially $\det(AB) = \det(A)\det(B)$

Linear Systems

… again

Linear Systems in Matrix Form

- Consider the linear system

$$\begin{vmatrix} 2x - 5y + 3z &= -3 \\ 4x + 0y + 8z &= 0 \\ 1x + 3y + 0z &= 2 \end{vmatrix}$$

- Unknown variables x, y, z
- We can represent this as a matrix equation

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

  - Or more generally, $Ax = b$
  - Looks like a linear equation "on steroids"
- Transformation perspective
  - The matrix A specifies a transformation
  - This transformation was applied to $[x\ y\ z]^T$ to get $[-3\ 0\ 2]^T$
  - Can we find the **inverse transformation**?

Inverse Matrix

- Consider a general, “good” transformation
  - We’ll talk later which transformations are not so good
  - The inverse transformation will “bring back” the basis vectors
  - E.g. $90°$clockwise rotation $\Rightarrow 90°$ counterclockwise rotation
- The inverse transformation has its own matrix: T^{-1}
- If we apply the transformation and the inverse, we’ll get our initial result
  - This means, nothing will change
  - In math terms, $T^{-1}T = E$
- Let’s now try to apply the inverse transformation to our linear system
  - Note that this means multiplying on the left

$$Ax = b$$
$$A^{-1}A = E \Rightarrow Ex = A^{-1}b$$
$$Ex = x \Rightarrow x = A^{-1}b$$

Solving the Linear System

- So, to find the unknown vector x, we need to find the inverse matrix of A – the matrix of the linear system
  - How?
  - There are many methods, the most popular of which is called Gaussian elimination (or Gauss – Jordan method)
  - We'll see this in the exercises
- Basic idea: $A^{-1}A = E$
  - Apply some transformation to get from A to E
  - Apply the same transformation to E
  - What we get is the inverse matrix
- Uses: a lot
  - Makes solving linear systems very fast and robust
  - Allows us to solve multiple linear systems with the same coefficients and different "free terms"

How About “Bad” Matrices?

- If the determinant of the matrix is zero, every element is mapped into a lower-dimensional space
  - 2D case: $\det(A) = 0 \Rightarrow$ every possible point in the 2D space is mapped onto a single vector
    - We can use the visualization to confirm this
    - We can also use our intuition: when is the area equal to zero?
  - Another viewpoint: $\det(A) = 0 \Rightarrow$ linearly dependent rows
- Consequently, there are many points mapped on the same point
  - Consider projection as an example
- No "reversible" function maps many inputs to one output
  - The matrix, therefore, is non-invertible
  - A^{-1} does not exist!
- However, solutions to the matrix can sometimes exist
  - We found one such solution in the first lecture
  - We’ll talk about this in more detail in the exercises

Summary

- Vectors – geometric and algebraic perspective
  - Operations
- Matrices
  - Definition, properties, operations
- Linear transformations
- Linear systems

Questions?