Linear Algebra

Generalizing our understanding of vectors and coordinates

Yordan Darakchiev

Technical Trainer

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

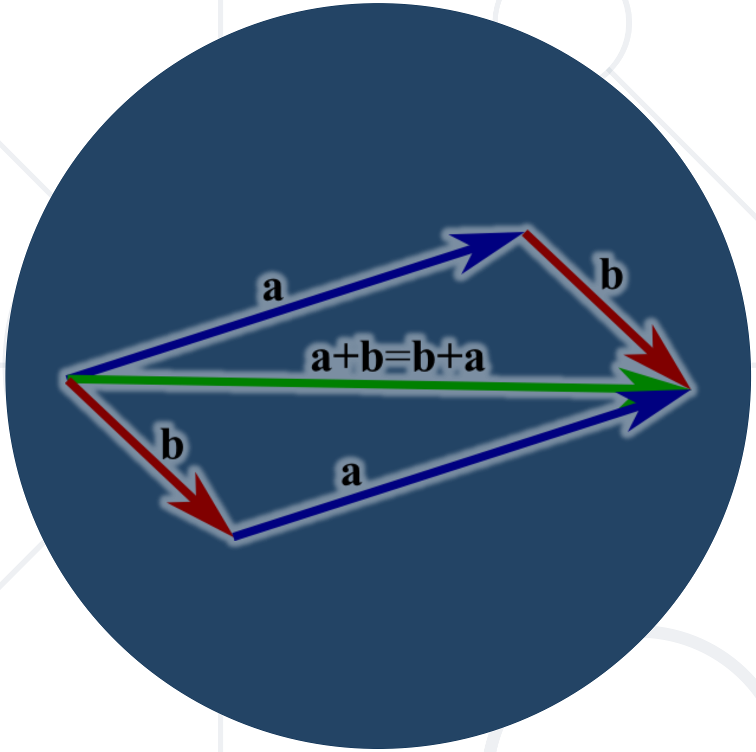
Software University

sli.do

#MathForDevs

Table of Contents

Software University

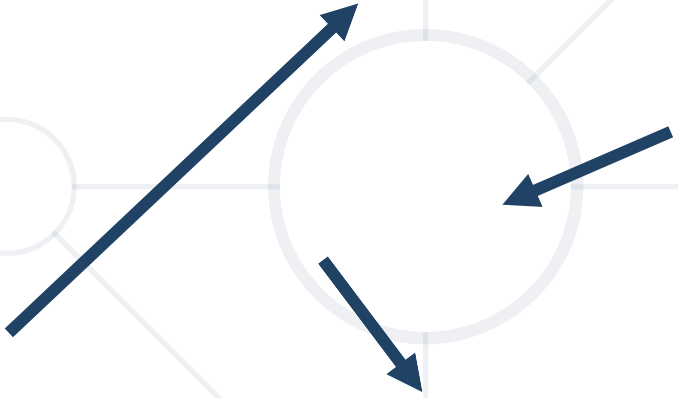


Vectors

Concrete and abstract

Definitions

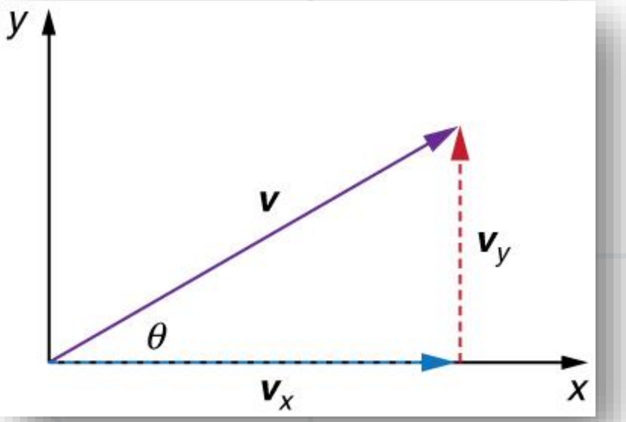
- **Physics definition**
  - A **pointed segment** in space
- **Computer science definition**
  - A **list of objects** (usually numbers)
  - Dimensions = length
- **Math definition**
  - Encompasses both, and allows **even more abstraction**: $\vec{v}$
  - Vectors can be **added** and **multiplied** by numbers and other vectors
  - Similar to **how we defined a field**



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

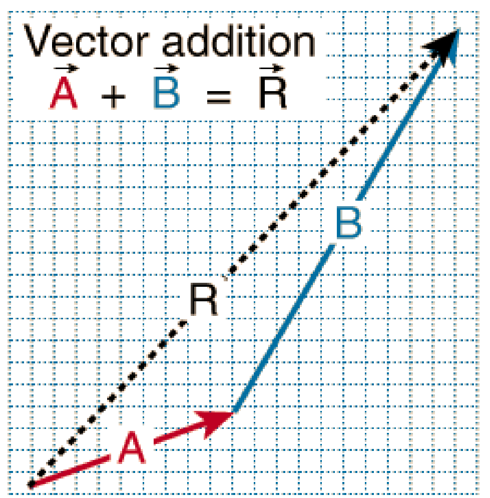
Components

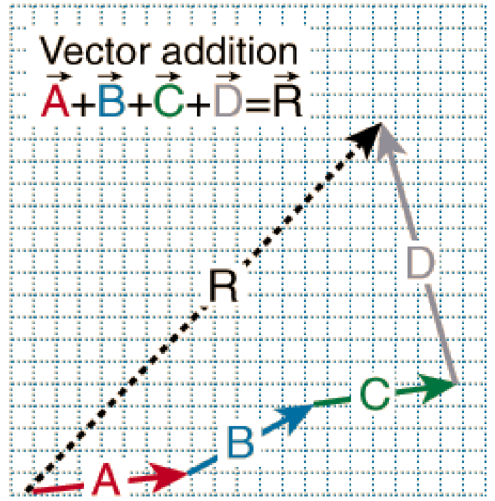
- The distances to all coordinate axes: v_x, v_y
- Equivalent to: $\begin{bmatrix} v_x \\ v_y \end{bmatrix}$
- Polar coordinates: $v = |\vec{v}|, \theta$
  - Finding components: $v_x = v\cos(\theta),\ v_y = v\sin(\theta)$
  - Finding the polar form: $v = \sqrt{v_x^2 + v_y^2},\ \theta = \tan^{-1}\left(\frac{v_y}{v_x}\right)$
- 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

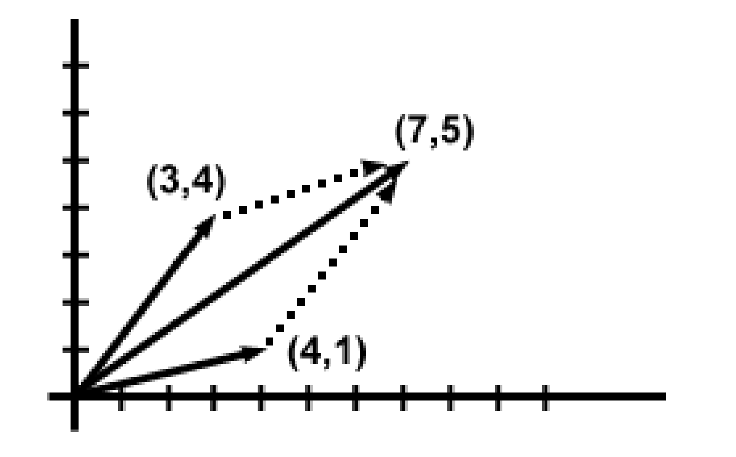


Operations

- **Addition**
  - Result:
    - **Length**: distance from start to end
  - **Direction**: start → end
  - In component form:
    sum components in each direction



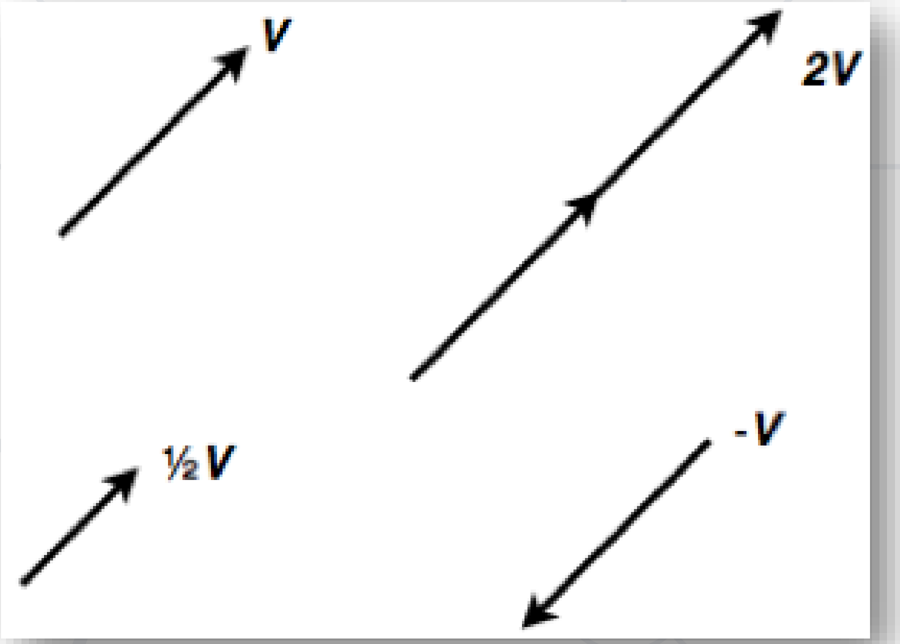




Operations (2)

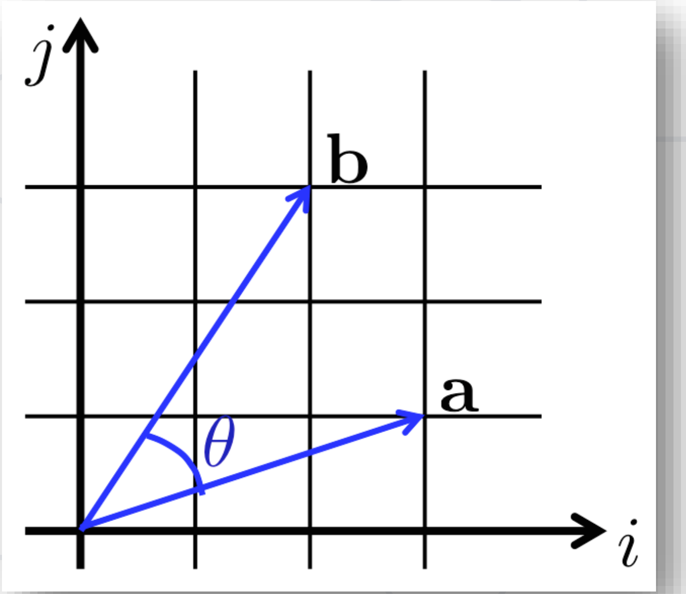
Software University

- **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**



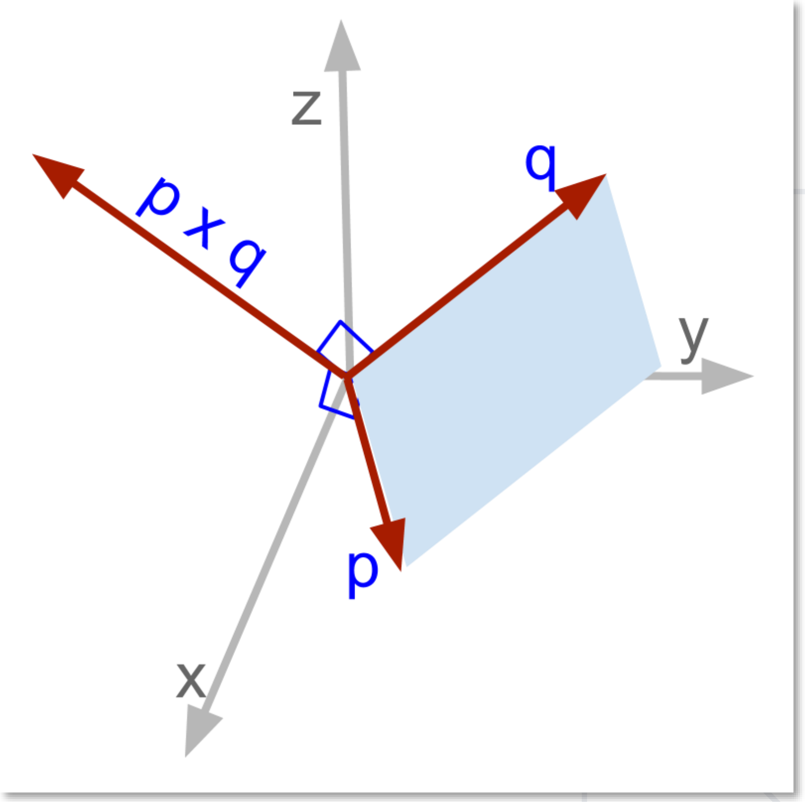
Operations (3)

- **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: $\vec{a}.\vec{b} = \sum_{i=1}^{n} a_i b_i$
  - Also defined by projecting one vector onto another (how?)



Operations

- **Vector product of two vectors (3D only)**
  - 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: $|\vec{a}|\left|\vec{b}\right| \sin(\theta)$ = area of parallelogram between $\vec{a}$ and $\vec{b}$
  - Direction: coincides with the direction of $\vec{n}$



Vector Spaces

Finding yourself in space

Definition

Software University

- 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

- **Coordinate space** (real / complex)
  - n-dimensional vectors
- **Infinite coordinate space**
  - Vectors with infinitely many components
- **Polynomial space**
  - All polynomials of variable x with real coefficients
- **Function space**
- **Matrix space** (stay tuned…)

Linear Combinations

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

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

- **Span** of a set of vectors
  - The set of all their linear combinations

$$\operatorname{Span}(V) = \left\{ \sum_{i=1}^{k} \lambda_i v_i \mid k \in \mathbf{N}, \lambda_i \in F, v_i \in V \right\}$$

Linear Combinations (2)

- **Linear (in)dependence**
  - The vectors $v_1, \dots, v_n$ 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 $\lambda_1 = 0, \lambda_2 = 0, \dots, \lambda_n = 0$
  - Conversely, they are **linearly dependent** if there is a non-trivial linear combination equal to zero
- Example:

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

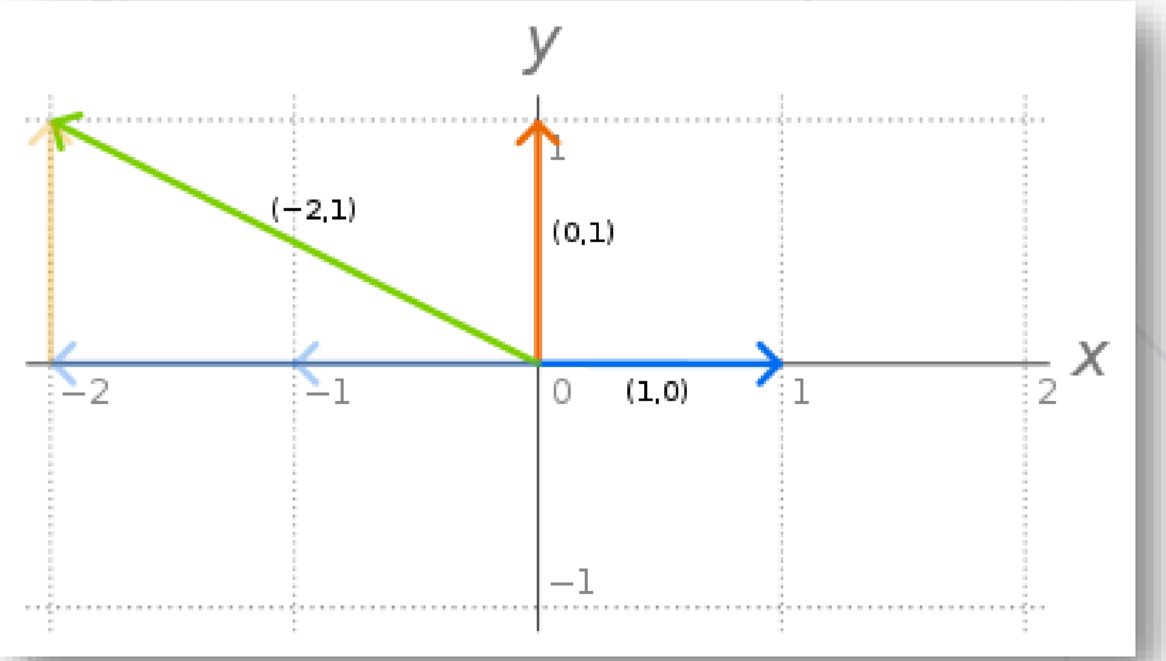
$$w = -2u + 3v \Rightarrow 2u - 3v + 1w = 0$$

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 (2)

- **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
  - Each set of n **LI vectors forms a basis in n-dimensional vector space**

$$
\begin{array}{c c}
 & \begin{array}{cccc} 1 & 2 & \cdots & n \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array} &
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
a_{31} & a_{32} & \cdots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\end{array}
$$

Matrices

Definition

- A **rectangular table of numbers**
- Dimensions: **rows × columns**
- 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
  - For almost all purposes it's OK to think about them as multidimensional matrices

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**
  - You may check this

Operations (2)

- 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**: $A_{m\times p}B_{p\times n} = C_{m\times n}$
- Definition: $c_{ij} = \sum_{k=1}^{p} a_{ik}b_{kj}$
- Example:

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

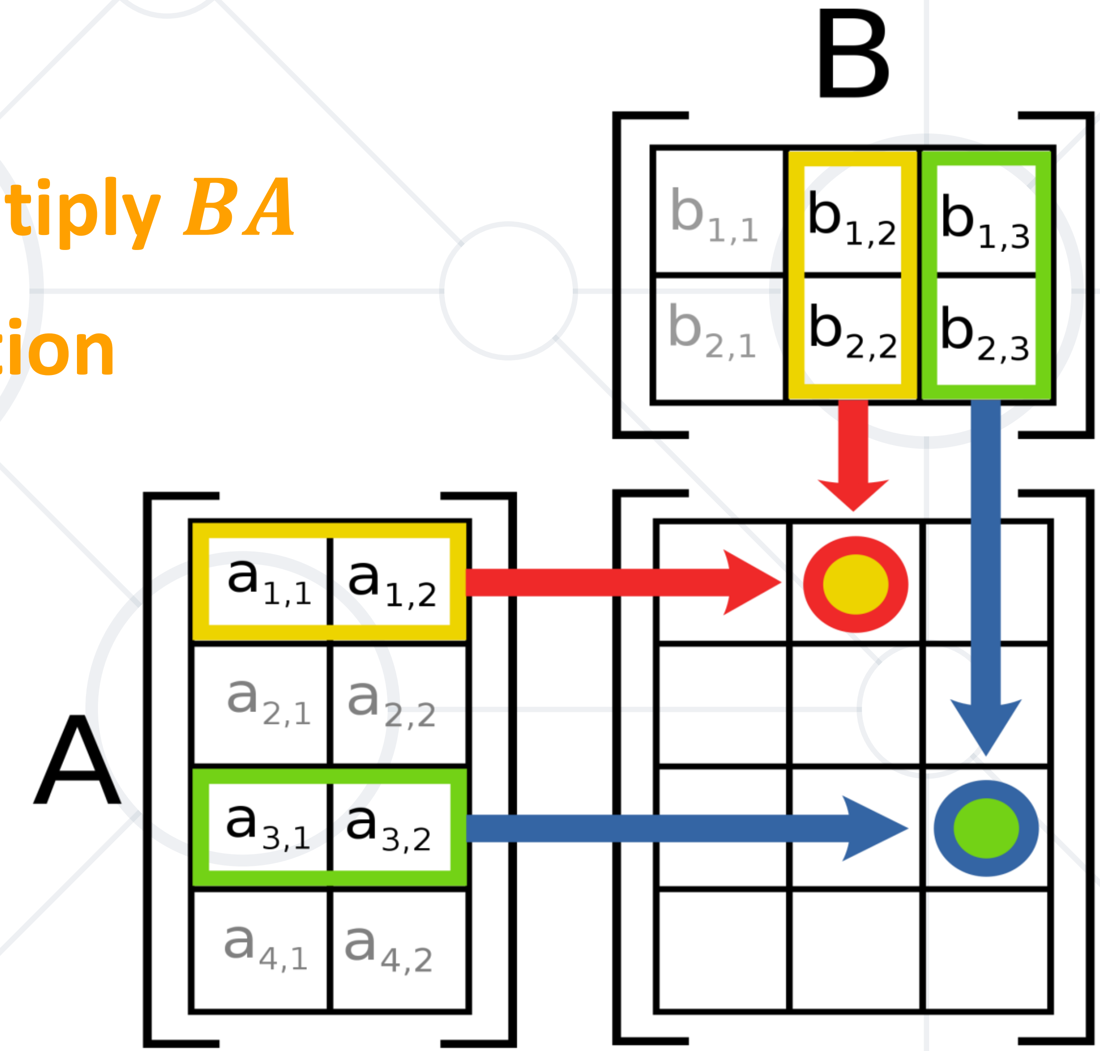
$$AB = \begin{bmatrix} 2.1 + 3.(-3) + 7.2 & 2.2 + 3.(-4) + 7.0 & 2.0 + 3.1 + 7.1 & 2.1 + 3.3 + 7.1 \\ 8.1 + 9.(-3) + 1.2 & 8.2 + 9.(-4) + 1.0 & 8.0 + 9.1 + 1.1 & 8.1 + 9.3 + 1.1 \end{bmatrix}$$

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

Matrix Multiplication (2)

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

      5.3 = 3.5 → commutative



Matrix Operations in numpy

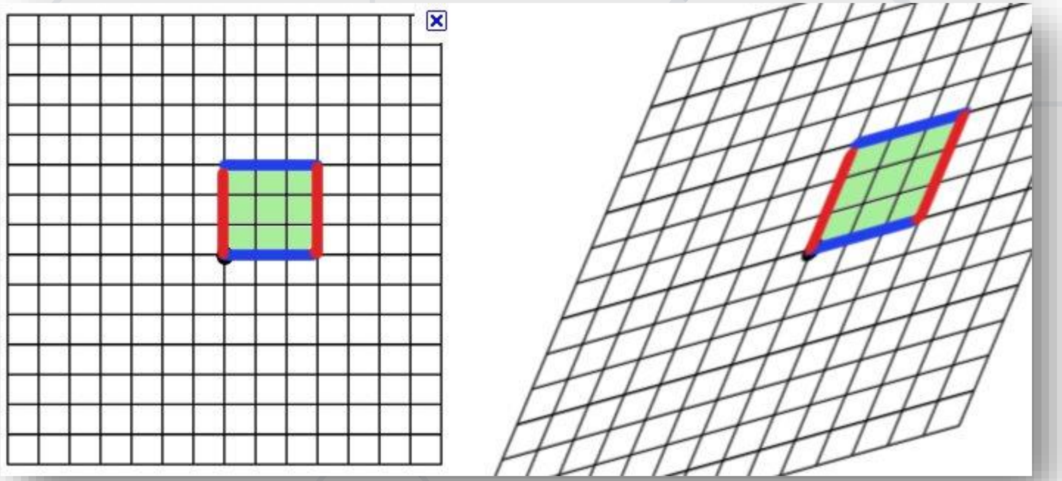
- We can use @ or `dot()` for both matrix multiplication and dot products
- **Note:** Whenever possible, use **numpy** arrays instead of lists

```
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: shapes not aligned
print(A.dot(B.T)) # Matrix multiplication
```

Transformation

- A mapping (function) between two vector spaces: $V \to W$
- Special case: mapping a space onto itself: $V \to V$
  - This is called a linear operator
- Each vector of V gets mapped to a vector in W



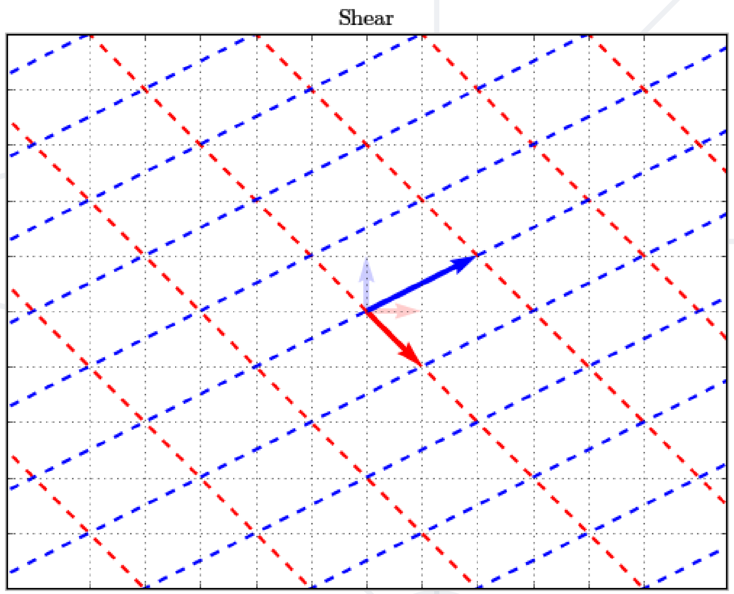
Linear Transformations

- 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: $v = v_x\hat{i} + v_y\hat{j}$ $\quad \begin{bmatrix} v_x \\ v_y \end{bmatrix} = v_x \begin{bmatrix} 1 \\ 0 \end{bmatrix} + v_y \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
  - New basis: $v' = v_x\hat{i}' + v_y\hat{j}'$ $\quad \begin{bmatrix} v'_x \\ v'_y \end{bmatrix} = v_x \begin{bmatrix} 1 \\ -1 \end{bmatrix} + v_y \begin{bmatrix} 2 \\ 1 \end{bmatrix}$
    - **Same coefficients, new basis vectors**
- This operation is called applying the linear transformation

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

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

- Applying the transformation to a vector is the same as multiplying the matrix times the original vector: $v' = Tv$
- Example: $T = \begin{bmatrix} 3 & 2 \\ -2 & 1 \end{bmatrix}, \ v = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$ $\quad v' = \begin{bmatrix} 3 & 2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 29 \\ -3 \end{bmatrix}$

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

    $\Rightarrow$ matrix multiplication $\equiv$ applying many transformations
- To visualize transformations, you can use the code in the `visualize_transformation.py` file

Multiple Transformations (2)

- 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 ☺
- This is especially useful in computer graphics

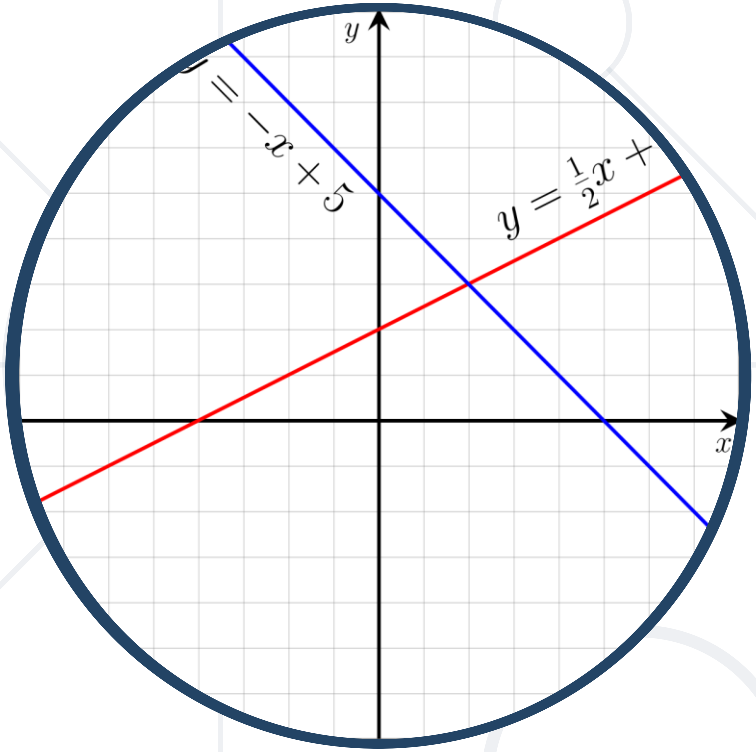
Multiple Transformations – Example

Software University

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

Determinant

- Measure of how much the unit area (volume) changes
- Scalar value
- Defined only for square matrices
- For more than two dimensions: area → volume
- The determinant of a matrix A is denoted $\det(A)$
- The determinant has very useful properties
  - Notably, $\det(AB) = \det(A)\det(B)$



Linear Systems

… again

Linear Systems in Matrix Form

- Consider the linear system

$$\left|\begin{array}{rcl} 2x - 5y + 3z & = & -3 \\ 4x + 0y + 8z & = & 0 \\ 1x + 3y + 0z & = & 2 \end{array}\right.$$

- 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"

Inverse Matrix

Software University

- Consider a **general**, "good" transformation
  - The inverse transformation will **"bring back" the basis vectors**
  - **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
  - I.e., nothing will change
  - In math terms: $T^{-1}T = E$

Inverse Matrix (2)

- 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 a Linear System

- To find the **unknown vector** x:
  - We need to find the **inverse matrix of** A
  - There are many methods, the most popular of which is called **Gaussian elimination** (or Gauss – Jordan method)
- **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**

Summary

- Vectors
  - Geometric and algebraic perspectives
  - Operations
- Matrices
  - Definition
  - Properties
  - Operations
- Linear transformations
- Linear systems

Software University

Questions?

SoftUni

- Software University
- SoftUni Creative
- SoftUni Digital
- SoftUni Foundation
- SoftUni Kids
- СофтУни БУДИТЕЛ
- Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License