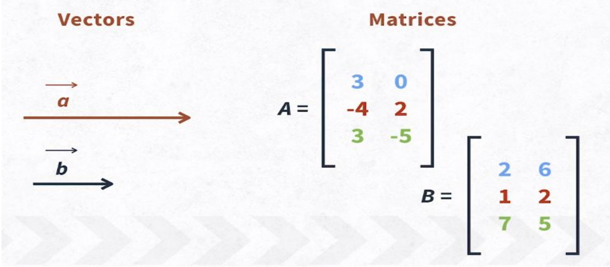
Linear Algebra

* Algebra is a branch of mathematics in which arithmetical operations and formal manipulation are applied to abstract symbols rather than specific number.

Linear Algebra:
* Main building blocks and areas are systems of linear equations , vectors and matrices, linear transformations,determinants, and vector spaces.
* Linear algebra is the study of vector and linear function.



- Dimensionality is called the length or the shape of the vector in python.

Scalar

*A number.Is denoted with a lowercase symbol, such as a or b.

*For example, weight,temperature,blood pressure

Vectors

*Lowercase bolded Roman letters

*Arrow print on top

*Vector is an ordered list of numbers

Vector Characteristics

*Dimensionality: the number of elements in the vector.

a = [1 9 6 5 4]

*Orientation: column orientation or row orientation.

*The Dimensionality is called the length or the shape of the vector in python.

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

Dimensionality: 9

Orientation

$$b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad c = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Example of Vectors

*x is a 3D column vector

*y is a 2D column vector

* z is a 3D row vector

* Orientation is important since the wrong orientation lead t
errors.

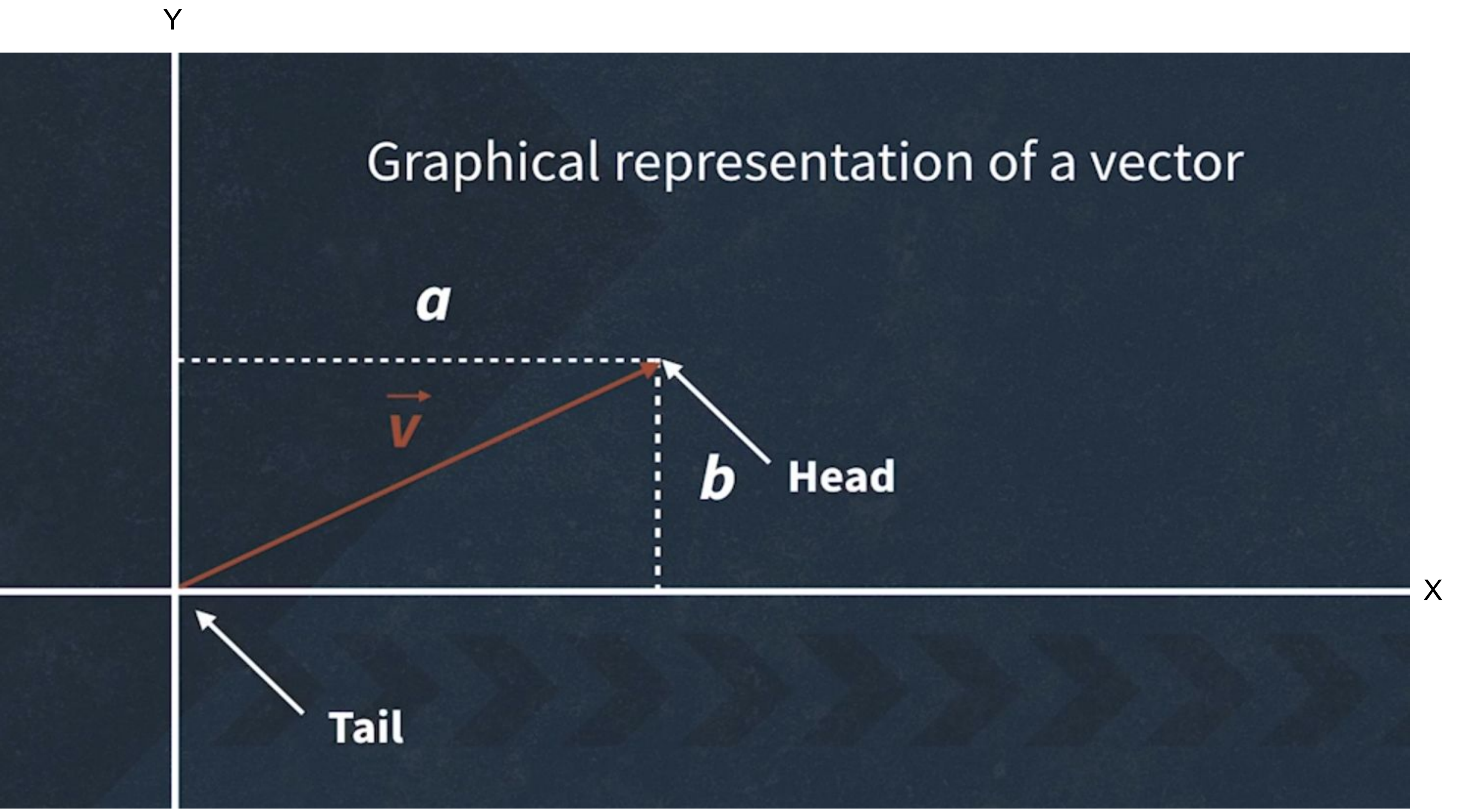
$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$z = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

- By convention, vectors are column oriented.
- Transpose operation converts a column vector into a row vector.

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$v^t = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$


Y
Graphical representation of a vector
a
$\vec{v}$
b
Head
X
Tail

Vector Representation

- VectorAsList = [1,2,3,4,5,6]
- VectorAsArray = np.array([1,2,3,4,5])

  -> This array is orientationless array : neither a row nor a column vector.

- In NumPy , we indicate orientation with brackets.

rowVector = np.array([[1,2,3,4,5]]) : The outer brackets just group all elements together in one object as an additional set of brackets indicates a row.

columnVector = np.array([[1],[2],[3],[4],[5],[6]]) : we see it has one column and 6 rows.

Vector Arithmetic

1. Vector Addition

- Add each corresponding element.

$$a = \begin{bmatrix} 3 & 5 & 5 & 2 & 4 \end{bmatrix}$$

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

$$a + b = ?$$

- Is possible only for two vectors that have the same dimension.

$$a = \begin{bmatrix} 3 & 5 & 5 & 2 & 4 \end{bmatrix}$$

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

$$a + b = \begin{bmatrix} 4 & 5 & 7 & 3 & 6 \end{bmatrix}$$

Vector Subtraction :

- Subtract each corresponding element.

$$a = \begin{bmatrix} 3 & 5 & 5 & 2 & 4 \end{bmatrix}$$

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

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

In [1]:
```
import numpy as np
```

In [10]:
```
#Vector Arithmetic
x = np.array([1,2,3,4])
y = np.array([5,6,7,8])
z = np.array([9,10,11,12])
w = np.array([20,21,22])
```

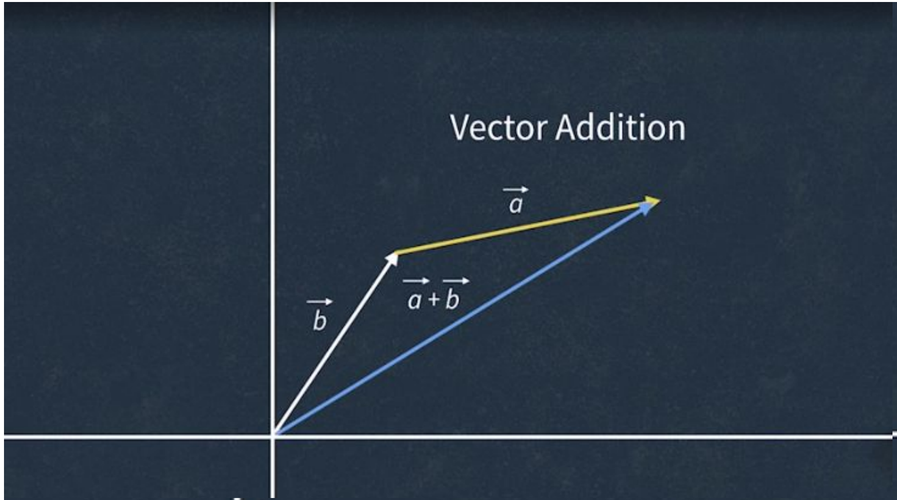
In [8]:
```
print("The addition of matrix:\n",x+y+z)
print("The subtraction of matrix:\n",x-y-z)
print("The multiplication of matrix:\n",x*y*z)
print("The division of matrix:\n",x/y)
```

```
The addition of matrix:
 [15 18 21 24]
The subtraction of matrix:
 [-13 -14 -15 -16]
The multiplication of matrix:
 [ 45 120 231 384]
The division of matrix:
 [0.2        0.33333333 0.42857143 0.5       ]
```

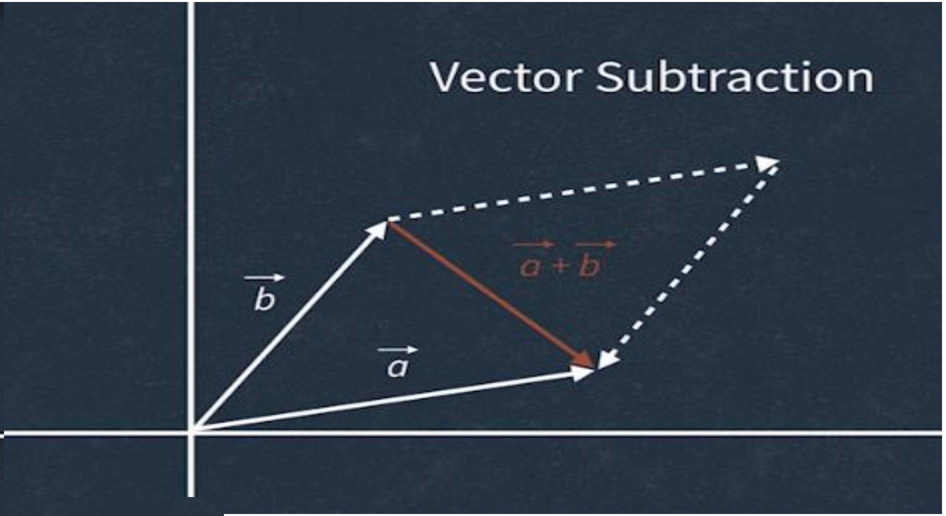
In [11]:
```
x+w # ca not be add  dimesion don't match
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
/tmp/ipykernel_22851/267012372.py in <module>
----> 1 x+w

ValueError: operands could not be broadcast together with shapes (4,)
(3,)
```

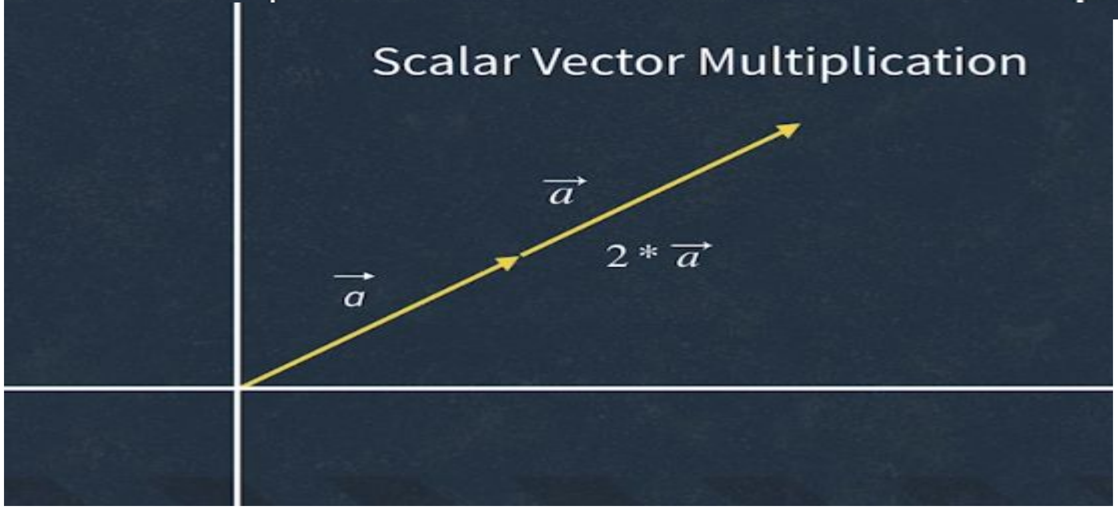
```
In [12]: #if we create a vector as a list or a vector as an ND array
         type_scaler = 3

In [13]: type_list = [10,20,30,40]
         print(type_list)

         [10, 20, 30, 40]

In [17]: type_array = np.array(type_list)
         print(type_array)

         [10 20 30 40]

In [18]: type_scaler * type_list

Out[18]: [10, 20, 30, 40, 10, 20, 30, 40, 10, 20, 30, 40]

In [19]: type_scaler * type_array

Out[19]: array([ 30,  60,  90, 120])
```



Vector Addition
$\vec{a}$
$\vec{b}$
$\vec{a}+\vec{b}$



Vector Subtraction
$\vec{b}$
$\vec{a}+\vec{b}$
$\vec{a}$



Scalar Vector Multiplication
$\vec{a}$
$2 * \vec{a}$
$\vec{a}$

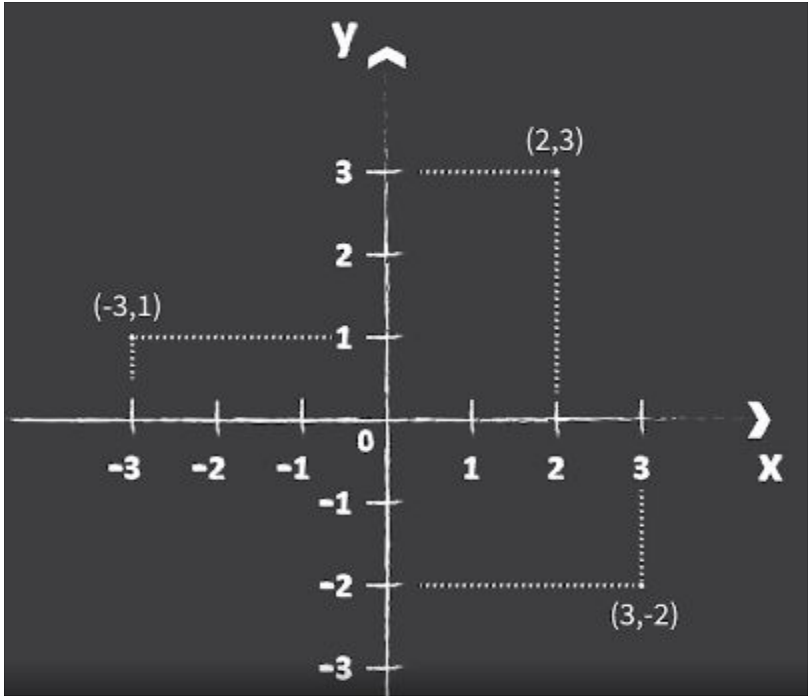
Coordinate System

- Describes where a certain position is located in a two-dimensional area.
- Coordinates have two numbers: the

  X-coordinate and the y-coordinate.
- The axes x and y meet at(0,0) coordinate

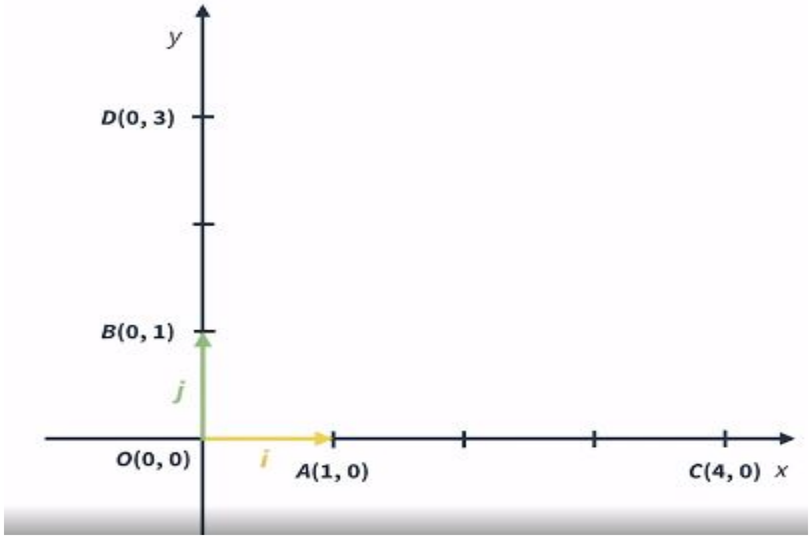
  Called the origin.
- Denoted by its distance along the x-axis,

  Followed by its distance along the y-axis.



- vec(OA) and vec(OB) vectors are called unit vectors along the x- and y-axes.
- They both have magnitudes that are equal to 1
- Eg. vec(OC) + vec (OD) = 4i + 3J
- Rule fr vector addition by placing the head of vector vec(OD) at the tail of the vector vec(OC).

  (vec = vector)



Unit or Basis Vectors

- Three properties of basis vectors:

  - Are linearly independent of each other

  - Span the whole space

  - Aren’t unique

Vector Projections and Basis

- Dot Product of Vectors

  - Three different ways it can be represented with symbols:

  - a^T b

  - a . b

  - <a, b>

  - Formula

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

Formula: $a \cdot b = \sum_{i=1}^{n} a_i b_i$

$$\overrightarrow{a} = [1,2,3,4,5]$$

$$\overrightarrow{b} = [6,7,8,9,10]$$

$$a \cdot b = 1 \cdot 6 + 2 \cdot 7 + 3 \cdot 8 + 4 \cdot 9 + 5 \cdot 10 = 130$$

```
In [2]: a = np.array([1,2,3,4,5])
        b = np.array([10,11,12,13,14])

In [3]: np.dot(a,b)

Out[3]: 190
```

Basic Properties of Dot Product

- It is commutative : a.b = b.a
- It is distributive: a(b+c) = a . b + a . c

- It is commutative : a . b = b . a

- It is distributive: a(b+c) = a.b + a.c

In [2]:
```
a = np.array([1,2,3,4,5])
b = np.array([10,11,12,13,14])
```

In [3]:
```
np.dot(a,b)
```

Out[3]: 190

In [6]:
```
c = np.array([20,21,22,23,24])
```

In [7]:
```
np.dot(b,a)
```

Out[7]: 190

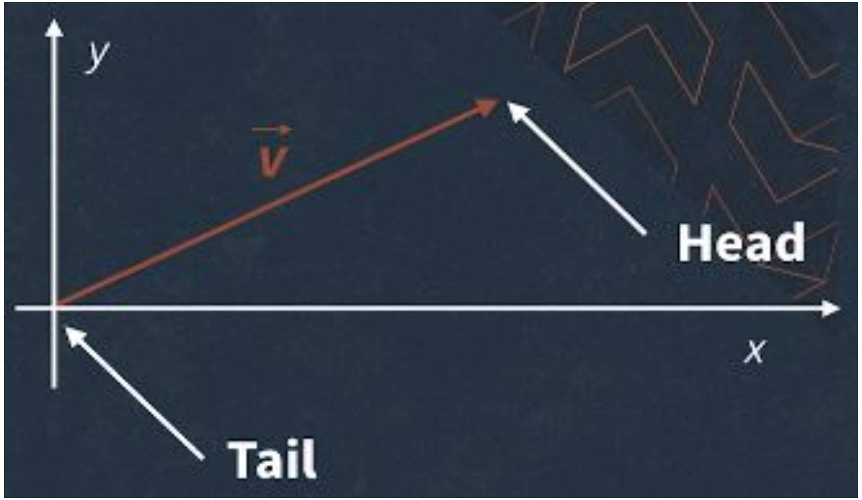
In [8]:
```
first_result=np.dot(a,b+c)
second_result=np.dot(a,b)+np.dot(a,c)

print(first_result)
print(second_result)
```

```
530
530
```

Scalar and Vector Projection

Magnitude of a Vector

- Also called norm or the geomatic length; is the distance from tail to head of a vector.



Magnitude of a Vector

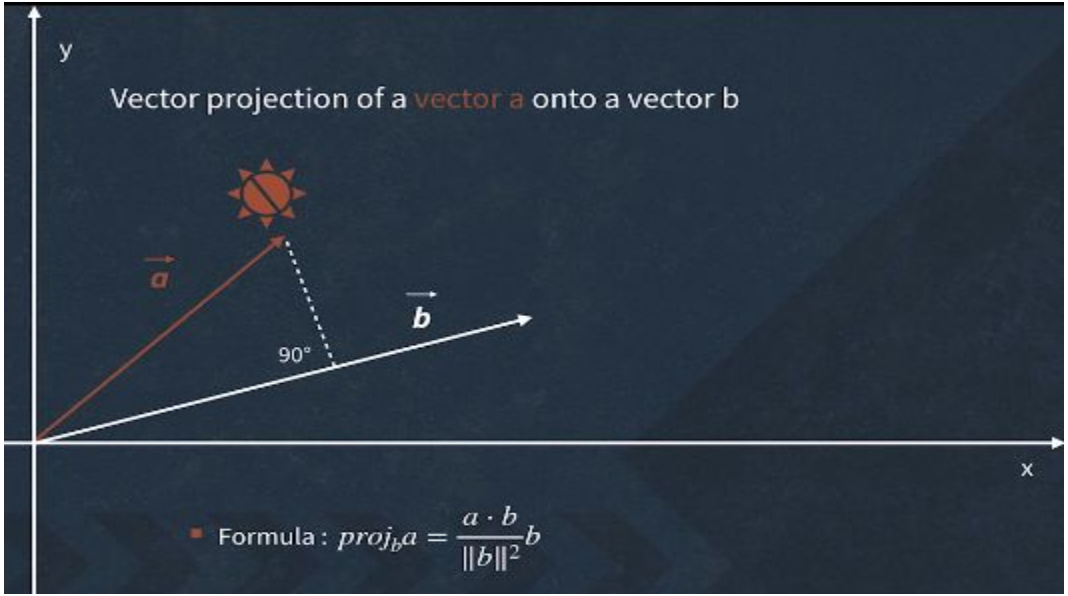
- We use double vertical bars around the vector ||x||.
- Formula :

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- There is a function in NumPy called norm
- Magnitude = np.norm(a)

Vector Projection

A vector projection of a vector a onto another vector b is the orthogonal projection of a onto b.



In [9]:
```
import numpy as np
from numpy import linalg as lng
```

In [10]:
```
a = np.array([1,2,3,4,5])
b = np.array([6,7,8,9,10])
```

In [11]:
```
lng.norm(a)
```

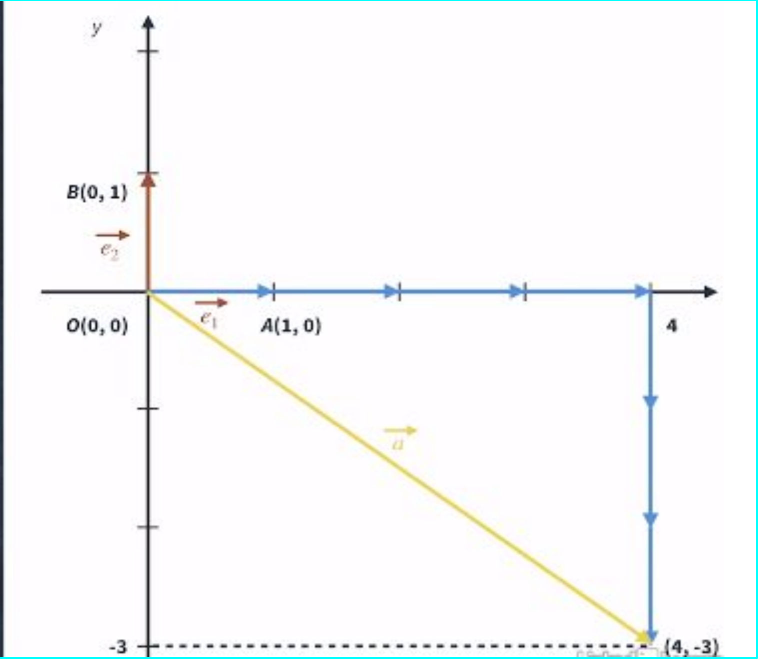
Out[11]:
```
7.416198487095663
```

In [12]:
```
vec_projection = (np.dot(a,b)/np.dot(b,b))*b
vec_projection
```

Out[12]:
```
array([2.36363636, 2.75757576, 3.15151515, 3.54545455, 3.93939394])
```

Changing Basic of Vectors

- e_1 and e_2 vectors are called unit vectors along the *x*- and *y*-axes
- a = $4e_1 - 3e_2$
- a = [4,-3]
- Unit vectors or basis vectors form a basis for space
- Any vector in space can be written as a linear combination of these two vectors



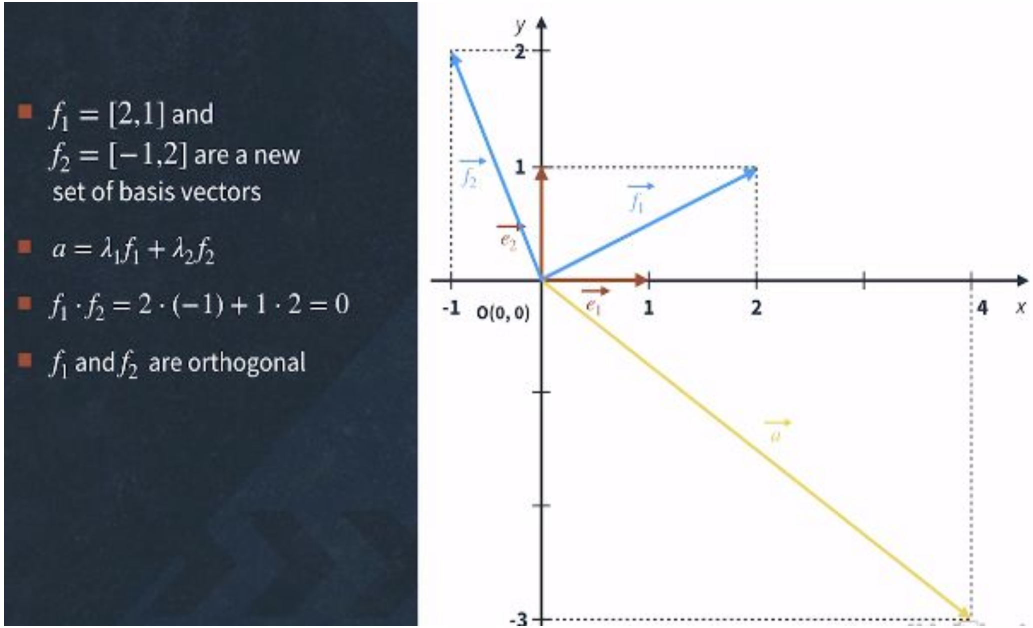
Unit or Basic Vectors

- Three properties of basis vectors:

  -Are linearly independent of each other.

  - Span the whole space

  - Aren’t unique



Vector projection of a vector onto a vector f1

Vector projection of a vector a onto a vector f_1

- Formula : $proj_{f_1} a = \frac{a \cdot f_1}{||f_1||^2} f_1$
- $proj_{f_1} a = \frac{4 \cdot 2 + (-3) \cdot 1}{2^2 + 1^2} [2,1]$
- $proj_{f_1} a = \frac{5}{5} [2,1] = 1 \cdot [2,1] = [2,1]$

Vector projection of a vector onto a vector f2

Vector projection of a vector a onto a vector f_2

- Formula : $proj_{f_2} a = \frac{a \cdot f_2}{\|f_2\|^2} f_2$
- $proj_{f_1} a = \frac{4 \cdot (-1) + (-3) \cdot 2}{(-1)^2 + 2^2} [-1,2]$
- $proj_{f_1} a = \frac{-10}{5} [-1,2] = -2 \cdot [-1,2] = [2, -4]$

Vector a can be written as:

- $a = [4, -3]$
- $a = 1 \cdot [2,1] + (-2) \cdot [-1,2]$

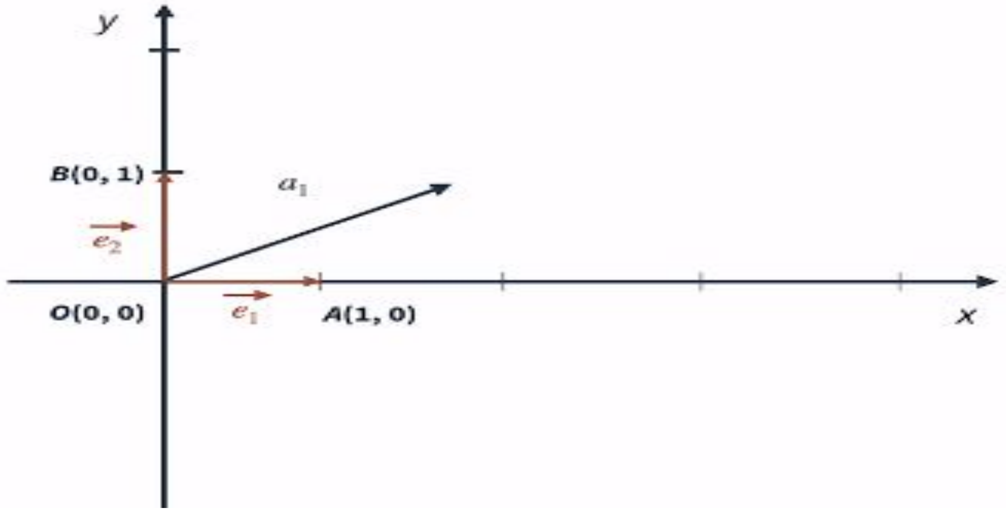
Basis, Linear Independence, and Span

Spanning Set

- The set {V1,...Vn} is a spanning set for V if and only if every vector in V can be written as a linear combination of V1,...,Vn.

- Span consists of all vectors of the form $\lambda_1 a_1$
- λ_1 can be positive, negative, or zero
- By taking a multiple of v_1, you can get anywhere along one-dimensional space of a line

- If we want to span the entire space, we'll need at last two vectors
- Any vector a in R^2 can be represented as a linear combination of e_1 and e_2, and hence $\{e_1, e_2\}$ is a spanning set for R^2
- $a = \lambda_1 e_1 + \lambda_2 e_2$


y
B(0, 1)
a_1
$\overrightarrow{e_2}$
O(0, 0)
$\overrightarrow{e_1}$
A(1, 0)
x

Exceptions

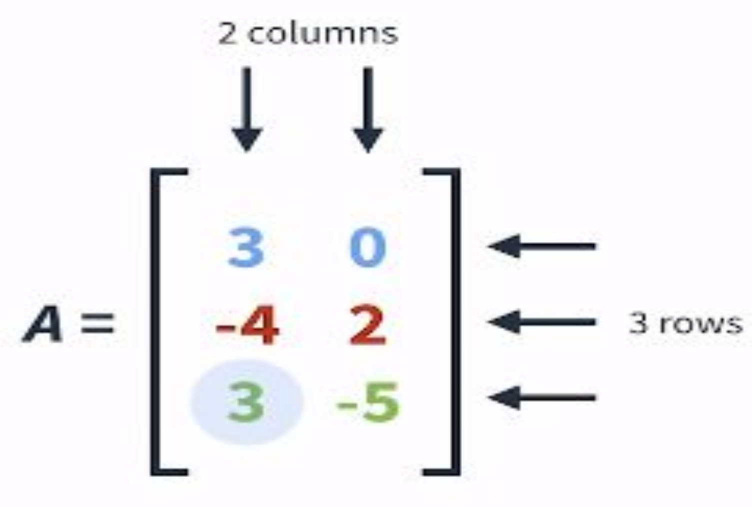
- Two vectors that line up in the same direction.
- These two vectors are null vectors.

FOR VECTORS to BECOME BASIS:

- They don’t have to be unit vectors, they can be any given length.
- They don’t have to be orthogonal , they don’t have to at 90 degree to each other.

Introduction to Matrices

- Collection of numbers ordered in rows and columns.
- Two-dimensional array of numbers
- Denoted matrices in uppercase, italic and bold-for example, A.
- Each of these values is called an element.



- Basic arithmetic operations can be applied on matrices.
- Contains numbers, symbols, or expressions.
- Matrix are Any size.
- M by n matrix(means it has m rows and n columns).

$$A = \begin{bmatrix} 3 & 0 \\ -4 & 2 \\ 3 & -5 \end{bmatrix} \qquad B = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

$$C = \begin{bmatrix} 2x - 5 & 4y - 8 \\ -4x - 10 & 5y + 2z \end{bmatrix}$$

- a_{ij}, the element on the position i and j; i represents the row, and j represents the column
- Matrix A that has m by n elements

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ & \cdots & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

- In Python, arrays start from 0, so our matrix would begin with element a_{00} and end with element a_{m-1n-1}

$$A = \begin{bmatrix} a_{00} & \cdots & a_{0n-1} \\ a_{10} & \cdots & a_{1n-1} \\ \cdots & & \\ a_{m-10} & \cdots & a_{m-1n-1} \end{bmatrix}$$

Types of Matrices

1. Rectangular
2. Square
3. Symmetric
4. Zero
5. Identity
6. Diagonal
7. Triangular

Rectangular Matrix

- Is a matrix that has a different number of rows and columns
- It's an m by n matrix where m is the number of rows and n is the number of columns

2 columns

$$A = \begin{bmatrix} 3 & 0 \\ -4 & 2 \\ 3 & -5 \\ 2 & 7 \end{bmatrix}$$

4 rows

Square Matrix

- A special case of a rectangular matrix
- It has the same number of rows and columns
- It is denoted as m by m matrix

$$\begin{array}{ccc} & \text{3 columns} & \\ A = & \begin{bmatrix} 3 & 0 & 4 \\ -4 & 2 & 6 \\ 3 & -5 & 8 \end{bmatrix} & \text{3 rows} \end{array}$$

Symmetric Matrix

- A special type of square matrix that has elements mirrored across the diagonal
- All the corresponding mirrored elements are the same

$$\begin{array}{ccc} & \text{3 columns} & \\ A = & \begin{bmatrix} 1 & 5 & 7 \\ 5 & 2 & 9 \\ 7 & 9 & 8 \end{bmatrix} & \text{3 rows} \end{array}$$

Zero Matrix

- Is the matrix that has all elements equal to zero
- Any vector or matrix multiplied with a zero matrix will be equal to the zero matrix

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

Identity Matrix

- Is a square matrix that has all zeros on off-diagonal elements and all ones on the diagonal elements
- It is denoted with the capital letter I
- When we multiply any vector or matrix with the identity matrix, we'll get the same vector or matrix

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

Diagonal Matrix

- Is a matrix in which all off-diagonal elements are equal to zero
- Diagonal elements can be any numbers, zeros included
- When we multiply any scalar with the identity matrix, we'll get a diagonal matrix

$$D = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

Triangular Matrix

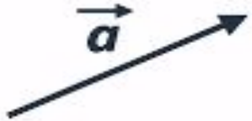
- Is a square matrix that has elements on the upper right or the lower left of the matrix equal to zero
- An upper triangular matrix has nonzero elements above the diagonal
- A lower triangular matrix has all zero elements above the diagonal

$$U = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

Types of matrix transformation

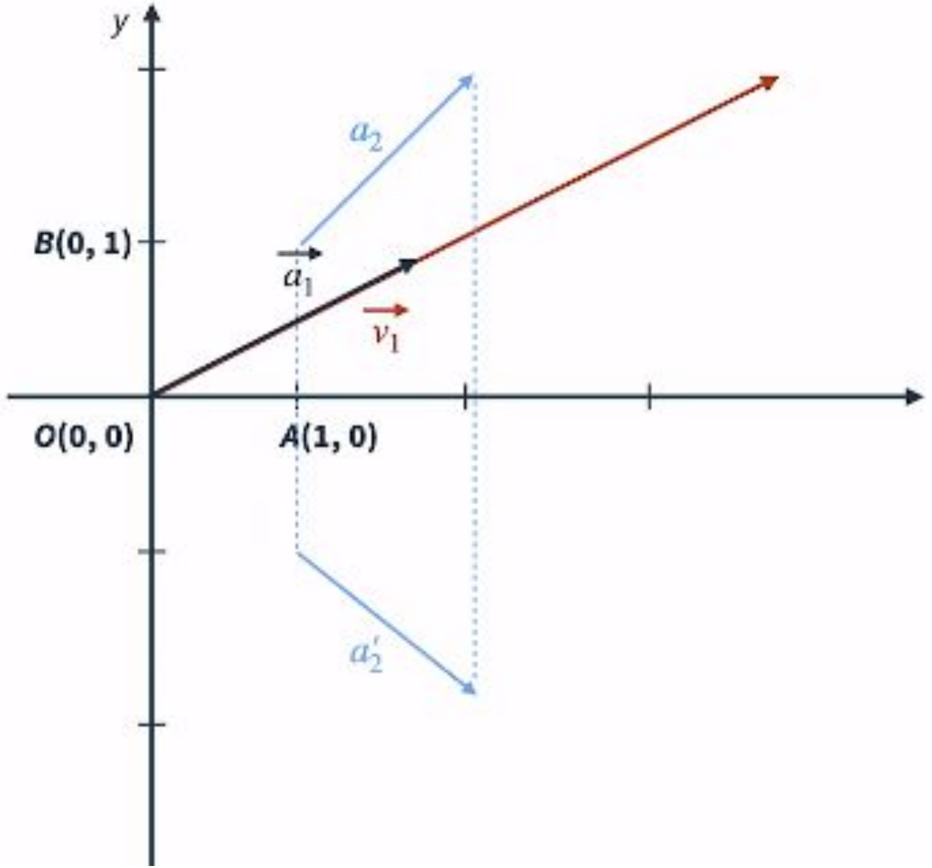
- Any linear transformation in a plane or in a space can be specified using vectors or matrices
- We can specify any linear transformation in three-dimensional space by using a matrix that has nine elements



$$A = \begin{bmatrix} 3 & 0 \\ -4 & 2 \\ 3 & -5 \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- Scaling by a factor in a direction
- Reflection across the plane
- Rotation by angle about any axis
- Projection onto any plane or some composition of transformations



- If we take an identity matrix and multiply it with vector [a,b], we get the same vector [a,b]

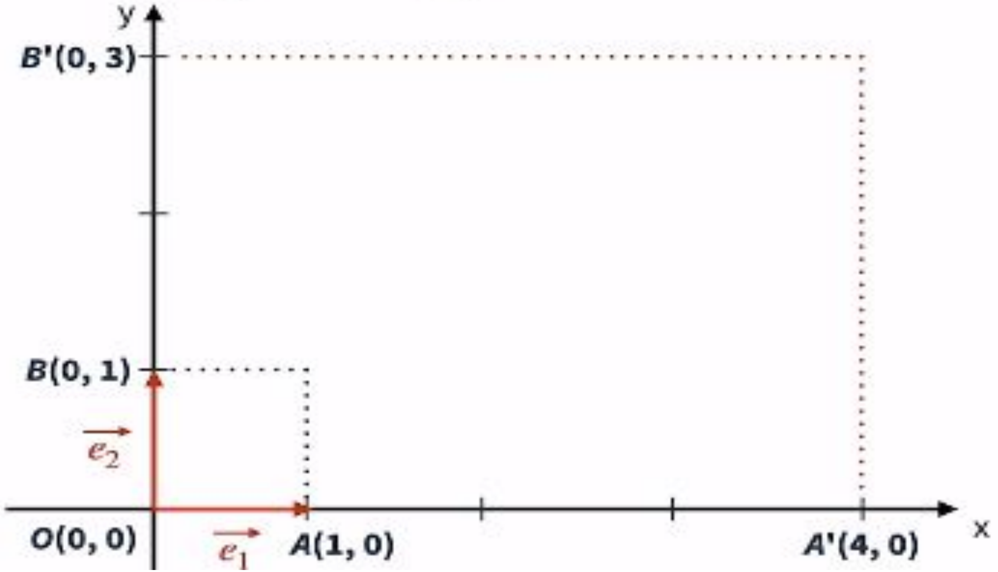
$$v = \begin{bmatrix} a \\ b \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I \cdot v = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

Scaling

- Diagonal matrix with scales *x*-axis by multiple of 4 and *y*-axis by multiple of 3
- A stretch from a square into a rectangle

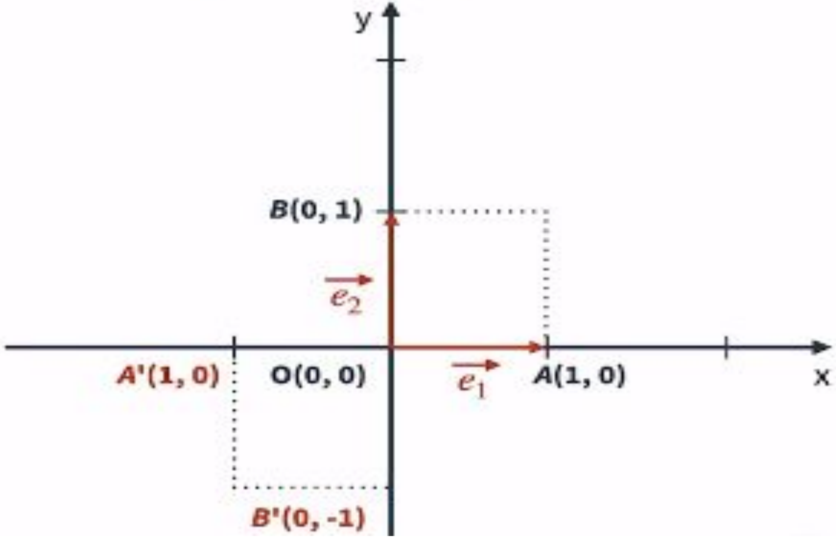
$$A = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix}$$



Inversion

- A diagonal matrix with elements -1,0,0,-1
- Flipped both *x*- and *y*-coordinates

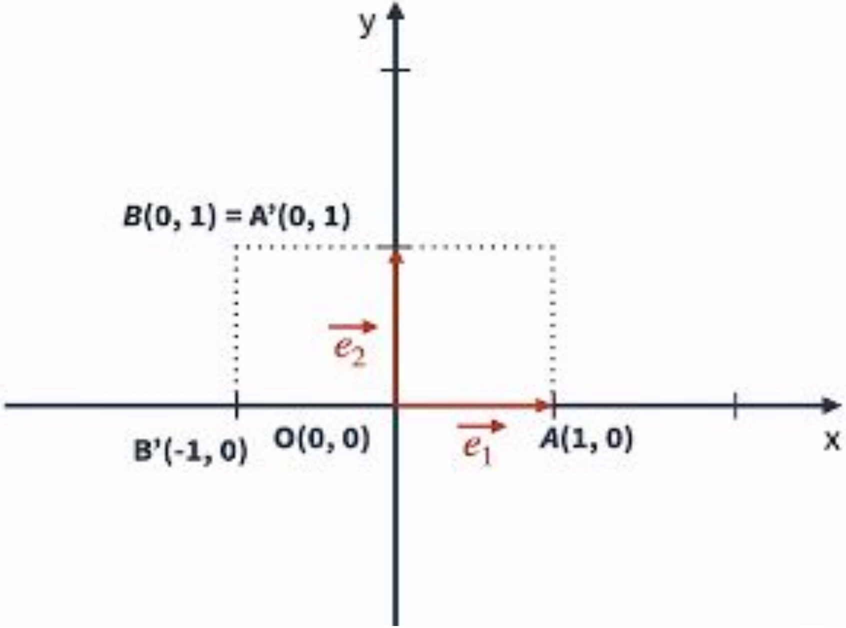
$$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$



Rotation

- Matrix that is used in this case is called rotation matrix
- In our case, we rotate basis vectors for 90 degrees

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

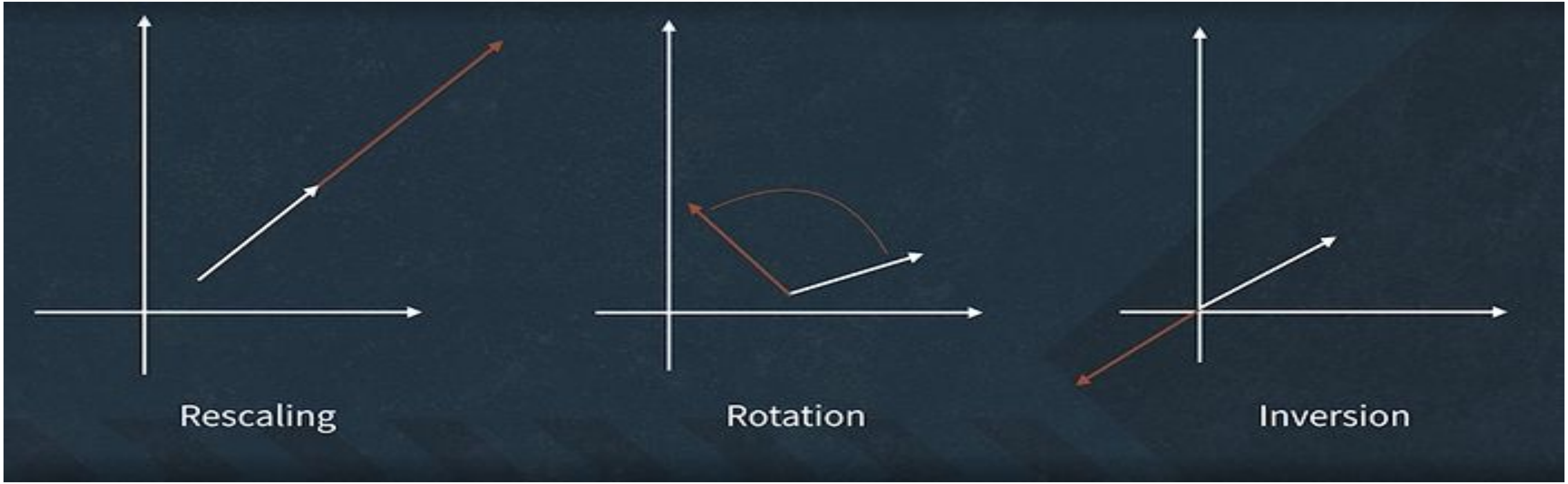


- Transformations that are combinations of two or more transformations- for example, stretching and rotation
- A transformation matrix with elements 4,1,-1,3, and when we multiply it with vector [1,2], we get an output vector [6,5]

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

$$A \bullet v = \begin{bmatrix} 4 & 1 \\ -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

Composition or Combination of Matrix transformations



Composition of Linear Transformations

Combining any number of different linear transformations to get a new linear transformation

- Any linear transformation can be represented with a matrix
- Any two or more composed linear transformations can also be represented as matrices
- Composition of two linear transformations A(B(v)), where A and B are matrices and v is the vector

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$v = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$A(B(v)) = A \cdot B \cdot v = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

How Do We Multiply Matrices?

- To multiply a matrix by another matrix, we need to do the dot product of rows and columns
- To calculate the values of the first element of AB, we have to multiply each corresponding element of the first row of A with each corresponding element of the first column of B and add the values

$$= A \cdot B = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} AB(e_1) & AB(e_2) & AB(e_3) \end{bmatrix}$$

- If we repeat this procedure, we find all elements of the product matrix
- At the end, we only have to multiply the matrix with a vector v
- The result of this calculation is the linear combination of the columns of the matrix AB with the coordinates x, y, and z as the scalars

Gaussian Elimination

- Solving Linear Equations Using Gaussian Elimination.

  -Karl Gauss : In the 1800s,German mathematician karl Gauss invented a method that is used for solving a system of linear equations.

  -That method is called Gaussian elimination.

- To create the augmented matrix, take the original matrix A and combine it with constant vector b
- Perform a series of elementary row operations on the augmented matrix, in a particular order
- Three possible situations: get the solution, system doesn't have a solution, system has an infinite number of solutions

$$
\begin{aligned}
x_1 + 2x_2 + x_3 &= 1 \\
2x_1 + -x_2 + x_3 &= 2 \\
4x_1 + 3x_2 + 3x_3 &= 4 \\
3x_1 + x_2 + 2x_3 &= 3
\end{aligned}
\rightarrow
\left[\begin{array}{ccc|c}
1 & 2 & 1 & 1 \\
2 & -1 & 1 & 2 \\
4 & 3 & 3 & 4 \\
3 & 1 & 2 & 3
\end{array}\right]
$$

$$
\begin{aligned}
x_1 &= \dots \\
x_2 &= \dots \\
x_3 &= \dots
\end{aligned}
$$

- The vertical line within the matrix indicates the separation between A and b
- Directly apply one or more row operations to the augmented matrix
- For row operations, you can do any arithmetic operation: add, subtract, multiply, or divide one of the rows with another row

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

$$\begin{array}{c} \\ + \\ : \\ - \end{array} \left[\begin{array}{ccc:c} 1 & 2 & 1 & 1 \\ 2 & -1 & 1 & 2 \\ 4 & 3 & 3 & 4 \\ 3 & 1 & 2 & 3 \end{array}\right]$$

Gaussian Elimination Can Be Broken into FIve Steps:

- Converting system to matrix-vector equation
- Augmenting the coefficient matrix with the vector of constants
- Creating a matrix with ones on diagonals.
- Mapping the matrix back to equations.
- Substitution to solve for variables.

- The first step begins by creating a coefficient matrix
- In the second step, we create a constant matrix
- By combining the coefficient matrix and constant matrix, we form an augmented matrix

$$\begin{aligned} 1x_1 + 2x_2 + 1x_3 &= 1 \\ 2x_1 - 1x_2 + 1x_3 &= 2 \\ 4x_1 + 3x_2 + 3x_3 &= 4 \\ 3x_1 + 1x_2 + 2x_3 &= 3 \end{aligned} \quad \longrightarrow \quad \begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 1 \\ 4 & 3 & 3 \\ 3 & 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

Coefficient Matrix

Constant Matrix

$$\left[\begin{array}{ccc:c} 1 & 2 & 1 & 1 \\ 2 & -1 & 1 & 2 \\ 4 & 3 & 3 & 4 \\ 3 & 1 & 2 & 3 \end{array}\right]$$

- Ensure that we have a nonzero entry in the diagonal position

$$\left[\begin{array}{ccc:c} 1 & 2 & 1 & 1 \\ 2 & -1 & 1 & 2 \\ 4 & 3 & 3 & 4 \\ 3 & 1 & 2 & 3 \end{array}\right]$$

Pivoting Step

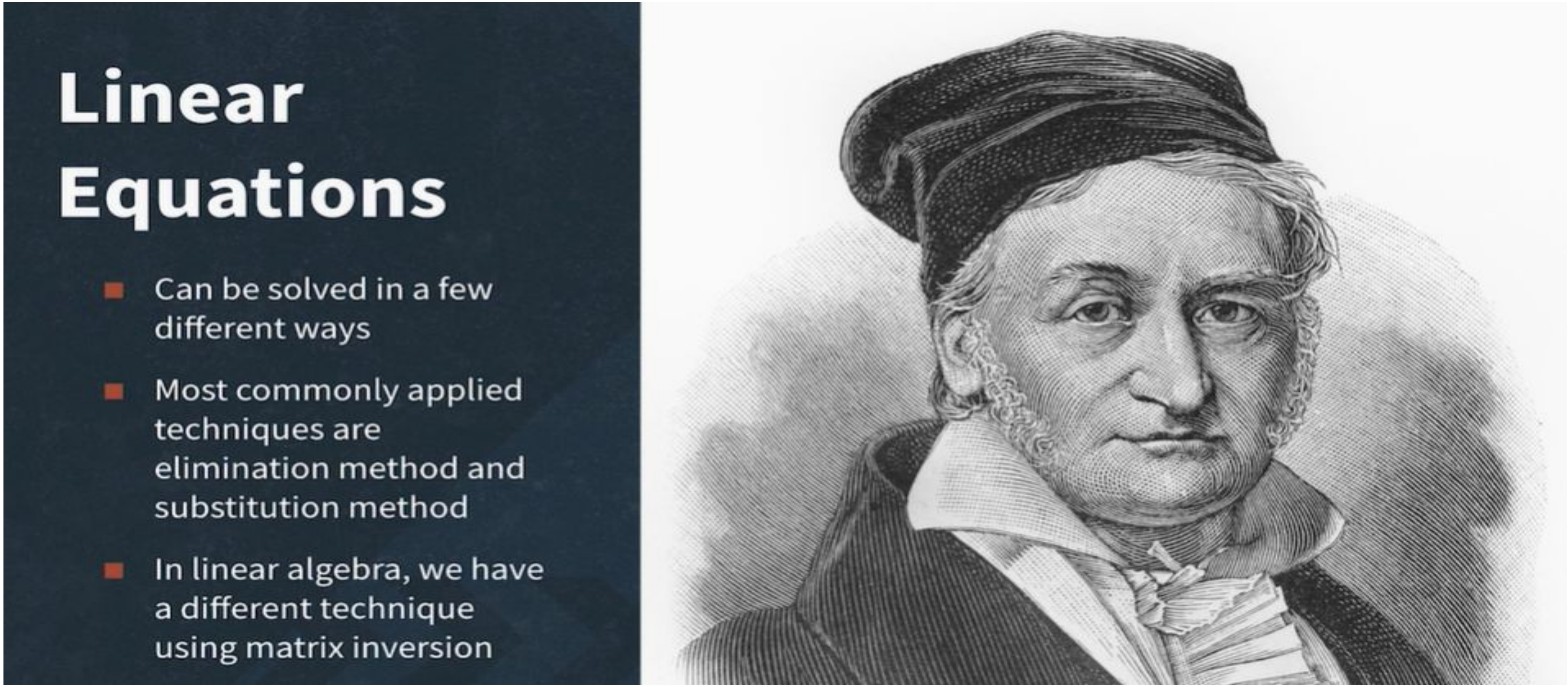
- Simple rules that we can follow:
  - Any two rows may be interchanged
  - Each row can be multiplied or divided by a nonzero constant
  - A nonzero multiple of one row can be added or subtracted to another row

- At the end of this process, our matrix is in the echelon form
- After transformation, our final matrix has three zeros in the third row, so we have eliminated the third row
- In the fourth step, we can map the matrix back to the equation
- In the last step, from the second row we get x_2, and when we substitute x_2 into the first equation, we get x_1

$$\left[\begin{array}{ccc:c} 1 & 2 & 1 & 1 \\ 2 & -1 & 1 & 2 \\ 4 & 3 & 3 & 4 \\ 3 & 1 & 2 & 3 \end{array}\right] \longrightarrow \left[\begin{array}{ccc:c} 1 & 2 & 1 & 1 \\ 0 & 1 & 0.2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$x_2 = -0.2x_3$$
$$x_1 + 2x_2 + x_3 = 1$$
$$x_1 = 1 - 2x_2 - x_3 = 1 - 0.6x_3$$

Gaussian Elimination and finding the inverse matrix



- The concept of inversion is well known starting from real numbers
- A real number a has a multiplicative inverse if there exists a number b such that a multiplied with b equals 1
- We can generalize the concept of multiplicative inverses to matrices
- Inverse of matrix A is denoted as A^{-1}

$$a \cdot b = 1 \longrightarrow A \cdot B = I$$

$$A^{-1}$$

$$A \cdot A^{-1} = I$$

$$A \cdot B = B \cdot A = I$$

- A matrix B such that $A \cdot B = B \cdot A = I$ is called an inverse matrix of the matrix A
- If matrix A is invertible and we want to solve the system, then for any vector b the system has a unique solution *x*
- We can get *x* by multiplying both sides of an equation on the left by the matrix A^{-1}

$$A \cdot B = B \cdot A = I$$

$$A \cdot x = B/ \cdot A^{-1}$$

$$\underbrace{A^{-1} \cdot A}_{I} \cdot x = A^{-1} \cdot B$$

$$\boxed{x = A^{-1} \cdot B}$$

Gaussian elimination and finding the inverse matrix

In [1]:
```python
import numpy as np
```

In [2]:
```python
A= np.array([[1,2],[3,4]])
A
```

Out[2]:
```
array([[1, 2],
       [3, 4]])
```

In [4]:
```python
Ainv=np.linalg.inv(A)
Ainv
```

Out[4]:
```
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
```

In [5]:
```python
b=np.array([5,11])
b
```

Out[5]:
```
array([ 5, 11])
```

In [ ]:
```python
x=np.dot(Ainv,b)
x
```

Out[5]:

array([5, 11])

In [6]:
```python
x=np.dot(Ainv,b)
x
```

Out[6]:
```
array([1., 2.])
```

In [7]:
```python
np.dot(A,x)
```

Out[7]:
```
array([ 5., 11.])
```

Inverse and Determinant

- A determinant for a matrix A is denoted as det(A)
- In the case when det(A) = 0, matrix A^{-1} cannot be computed
- In special cases matrix A is singular, meaning it contains only linearly dependent columns
- A determinant for a two-by-two matrix A can be computed using a simple formula:

$$det(A)$$

$$det(A) = 0 \longrightarrow \cancel{A^{-1}}$$

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

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

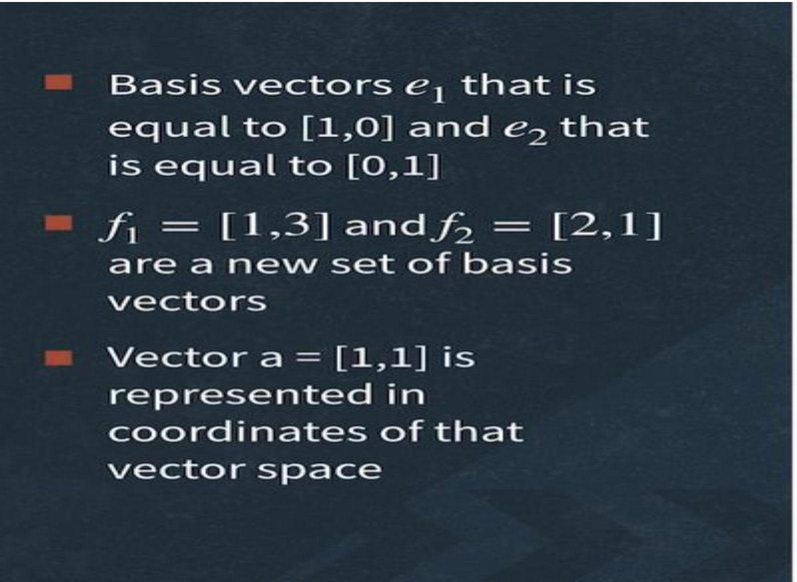
```
In [1]: import numpy as np

In [2]: A= np.array([[1,2],[3,4]])
        A

Out[2]: array([[1, 2],
               [3, 4]])

In [3]: det=np.linalg.det(A)
        det

Out[3]: -2.0000000000000004

In [4]: B= np.array([[3,1],[6,2]])
        B

Out[4]: array([[3, 1],
               [6, 2]])

In [5]: np.linalg.det(B)

Out[5]: 0.0
```

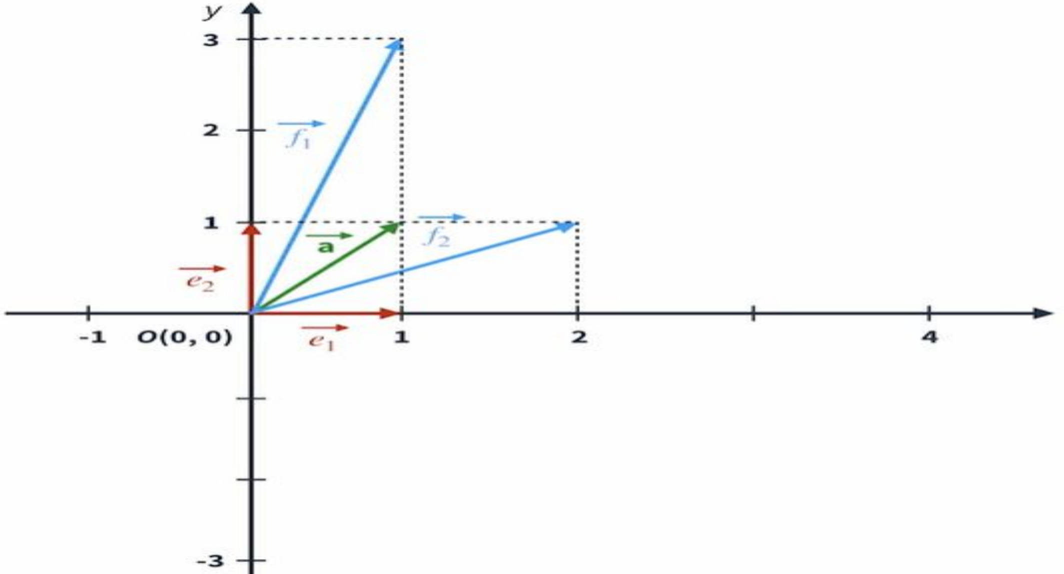
```
In [3]: det=np.linalg.det(A)
        det

Out[3]: -2.0000000000000004

In [4]: B= np.array([[3,1],[6,2]])
        B

Out[4]: array([[3, 1],
               [6, 2]])

In [5]: np.linalg.det(B)

Out[5]: 0.0

In [6]: np.linalg.inv(B)

---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-6-28ddc52c733a> in <module>
----> 1 np.linalg.inv(B)

<__array_function__ internals> in inv(*args, **kwargs)
```

Matrices from Orthogonality to Gram Schmidt Process

- Matrices Changing Basis.

  - A change of basis Matrix is a matrix that translates vector representations from one basic, such as the standard coordinate system, to another basis.





- If we take the matrix of new basis vectors f_1 and f_2 and multiply it with our vector a = [1,1], we get vector [3,4] as result
- A matrix constructed of new basis vectors f_1 and f_2 is called a transformation matrix A
- It represents the change of basis from the alternative vector space to standard vector space

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \quad a = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$A \bullet a = \begin{bmatrix} 1 \bullet 1 + 2 \bullet 1 \\ 3 \bullet 1 + 1 \bullet 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

- For a two-by-two matrix A, there is a simple formula to calculate A inverted

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \longrightarrow A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$$

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

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

- When we plug in the numbers in the formula, we get:

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

$$A^{-1} = \frac{1}{1 \cdot 1 - 2 \cdot 3} \begin{bmatrix} \mathbf{1} & \mathbf{-2} \\ \mathbf{-3} & \mathbf{1} \end{bmatrix}$$

$$A^{-1} = -\frac{1}{5} \begin{bmatrix} \mathbf{1} & \mathbf{-2} \\ \mathbf{-3} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} -\frac{1}{5} & \frac{2}{5} \\ \frac{3}{5} & -\frac{1}{5} \end{bmatrix}$$

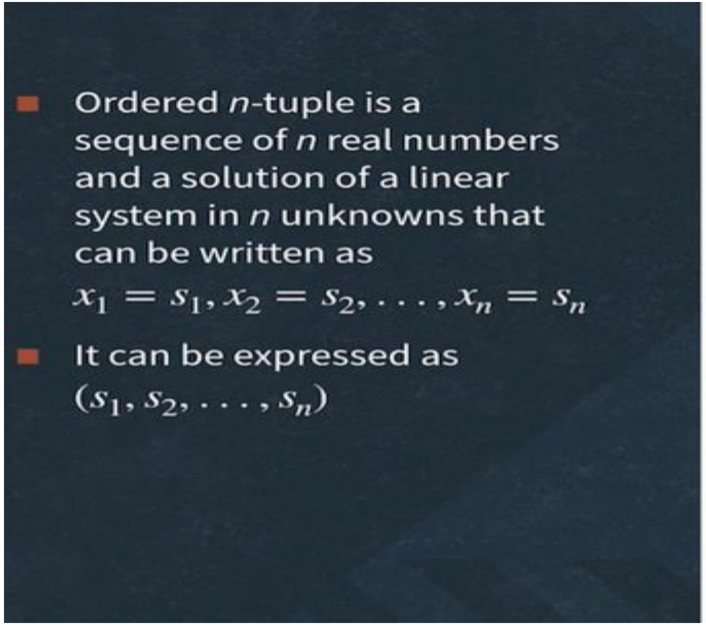
- Check our result by multiplying matrix A with matrix A^{-1}
- As you can see, we get the identity matrix as expected

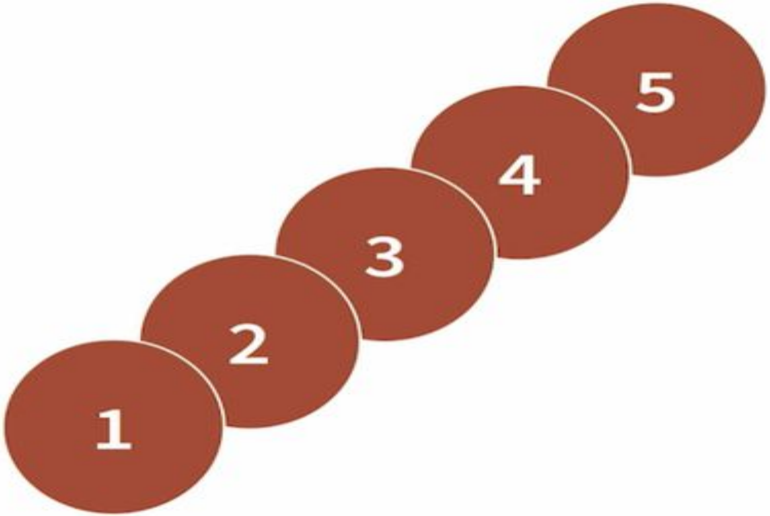
$$A \cdot A^{-1} = \begin{bmatrix} \mathbf{1} & \mathbf{-2} \\ \mathbf{-3} & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} -\frac{1}{5} & \frac{2}{5} \\ \frac{3}{5} & -\frac{1}{5} \end{bmatrix}$$

$$A \cdot A^{-1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \mathrm{I}$$

Matrix Transformations

- Matrix Transformations : are a special class of functions that arise from matrix multiplication.





- The set of all ordered *n*-tuples of real numbers is denoted with the bold capital letter R and superscript n: R^n
- The elements for R^n are called vectors
- Standard basis vectors are denoted as $e_1, e_2, \ldots, e_n$
- Every vector x can be written as
  $x = x_1e_1 + x_2e_2 + \ldots + x_ne_n$

- Matrix transformation from R^n to R^m:
  $T : R^n \rightarrow R^m$
- $y = Ax$
- Matrix transformation maps a vector x in R^n into the vector y in R^m by multiplying x with A
- It can be written as $y = T_a(x)$

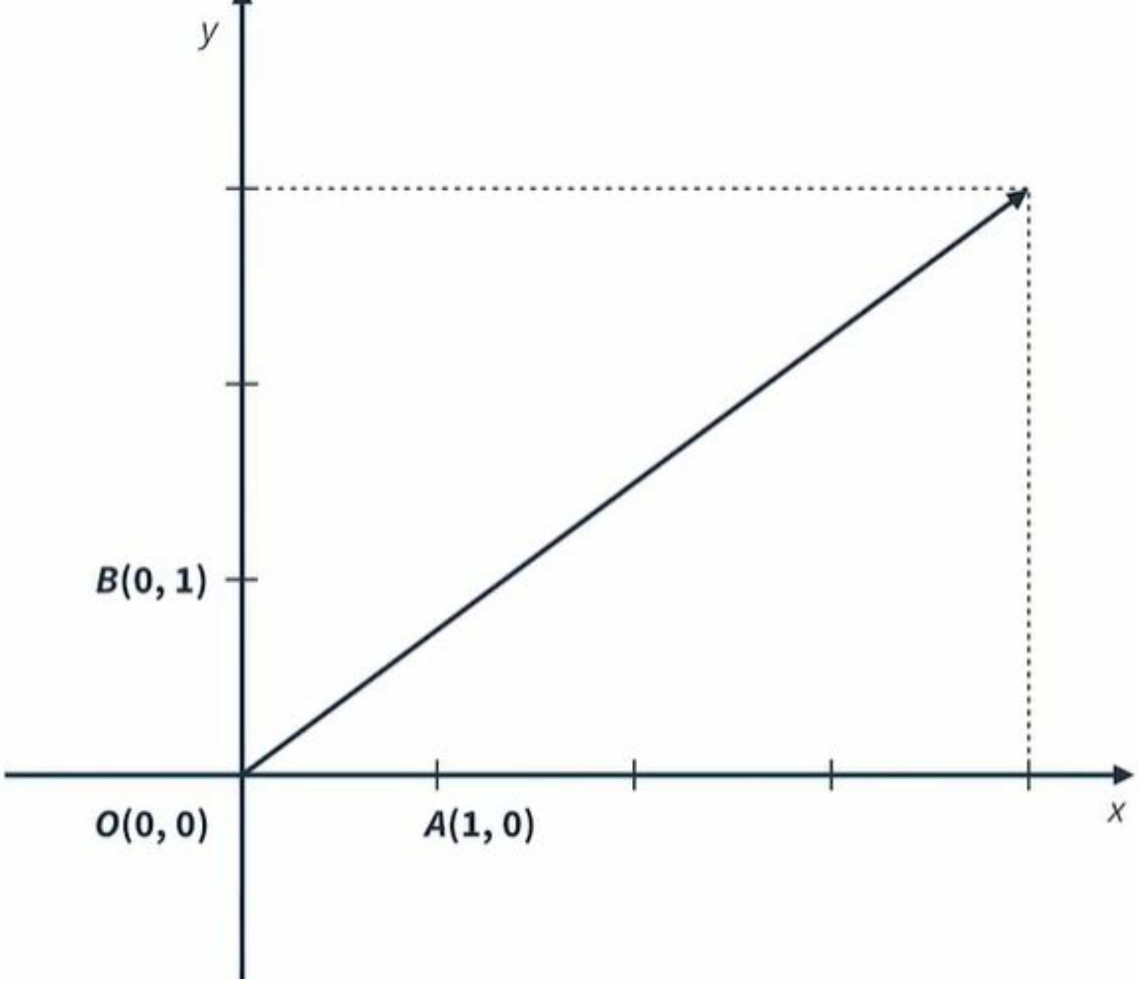
Transform a vector b to any basis with these steps:

- Transform the vector b to our standard coordinate system using the appropriate transformation matrix A that results in b': Ab = b'
- Perform a custom transform on b'
- Transformation is represented by the matrix R

- The standard coordinate system gives us a rotated vector c': Rb' = c'
- Transform c' back to the alternate coordinate system using the inverse of A that will result in the vector c
- $A^{-1}RA = R'$

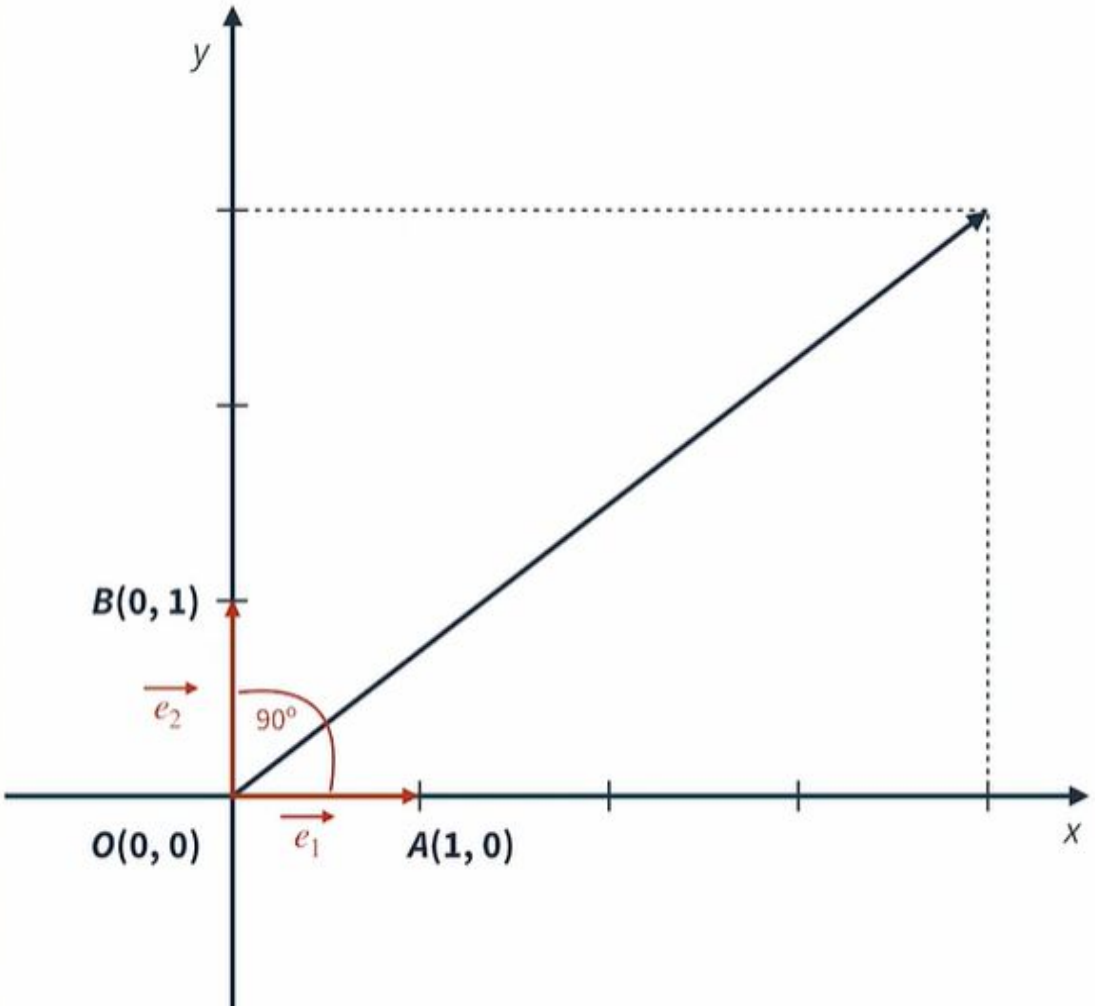
Standard Basis Vectors

- Are orthonormal, meaning they're orthogonal to each other



Standard Basis Vectors

- Are orthonormal, meaning they're orthogonal to each other
- They can be represented as vectors $e_1 = [1,0]$ and $e_2 = [0,1]$

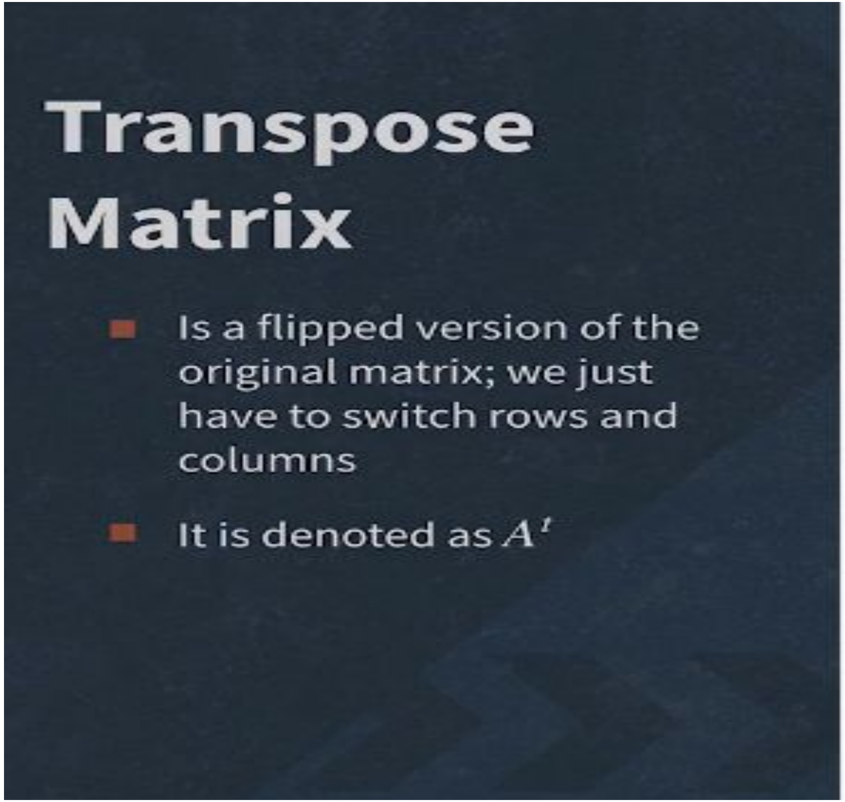


Orthogonal Matrix

- Is usually denoted with Q
- Orthonormal vectors make up all the rows and all the columns of the orthogonal matrix.

$$\mathbf{Q} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \cdots & & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Transpose Matrix



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

$$\downarrow$$

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

- An important property of orthogonal matrices is: when we multiply A^t with A, it is equal to the A multiplied with A^t, and they are both equal to the identity matrix
- When we calculate orthogonal matrices, we save computational time

$$A \cdot A^t = I$$

$$A^t \cdot A = I$$

Gram-Schmidt Process

- It is used to transform any basis to orthogonal basis
- Our matrix contains five columns
- Our first column stays the same
- We take the second column and orthogonalize it relative to the first column

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix}$$

- Apply the same process to the third column relative to the second column and to the first column
- Subtract off two parts: part of the column that is parallel to column two and part of the column that is parallel to column one
- Repeat the process until the last column

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix}$$

- At the end we get a matrix in which all the columns are orthogonal, but this matrix is not an orthogonal matrix
- In the next step, normalize each column
- Finally, get an orthogonal matrix that has all the columns pairwise orthogonal and they are all unit length

- B = $\{u_1, u_2, u_3\}$ is a basis for R^3
- The set B' = $\{v_1, v_2, v_3\}$ is an orthogonal basis for R^3

$$v_1 = u_1$$

$$v_2 = u_2 - \frac{u_2 \cdot v_1}{v_1 \cdot v_1} v_1$$

$$v_3 = u_3 - \frac{u_3 \cdot v_1}{v_1 \cdot v_1} v_1 - \frac{u_3 \cdot v_2}{v_2 \cdot v_2} v_2$$

- Plug in the values in our formulas
- v_1 is the same as u_1; we have to calculate v_2 and v_3

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

$$v_1 = < 1,1,1 >$$

$$v_2 = < 1,2,2 > - \frac{5}{3} < 1,1,1 > = < \frac{-2}{3}, \frac{1}{3}, \frac{1}{3} >$$

$$v_3 = < 1,1,0 > - \frac{2}{3} < 1,1,1 > + \frac{1}{2} < \frac{-2}{3}, \frac{1}{3}, \frac{1}{3} >$$

$$v_3 = < 1,1,0 > + < \frac{-2}{3}, \frac{-2}{3}, \frac{-2}{3} > + < \frac{-1}{3}, \frac{1}{6}, \frac{1}{6} >$$

$$v_3 = < 0, \frac{1}{2}, \frac{-1}{2} >$$

- Set B' is an orthogonal basis for R^3
- We need to normalize each vector in B' by calculating the norms for v_1, v_2, and v_3
- By dividing each vector with its norm, we get an orthonormal basis for R^3

$$w_1 = < \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}} >$$

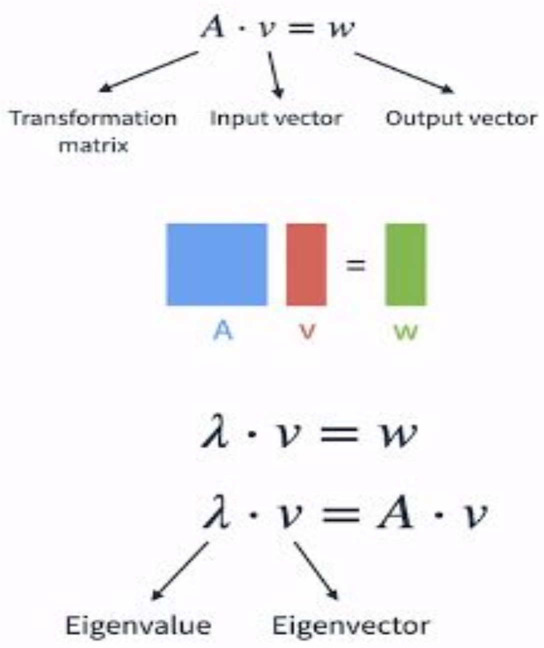
$$w_2 = < \frac{-2}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}} >$$

$$w_3 = < 0, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}} >$$

Introduction to Eigenvalues and EigenVectors

- Defined for square matrices
- Its goal is to extract pairs of eigenvalues and eigenvectors
- Each eigenvalue has an associated eigenvector

- If we apply some type of transformation on an input vector, we'll get an output vector
- Output vector w is a scaled representation of input vector
- Transformation matrix A is behaving like a single number, a scalar



- Let A be an $n \times n$ matrix
- A number λ is said to be an eigenvalue of A if there exists a nonzero solution vector K of the linear system $AK = \lambda K$
- The solution vector K is said to be an eigenvector corresponding to the eigenvalue λ

- Eigenvalues and eigenvectors are also called characteristic values and characteristic vectors, respectively
- Eigenvalues and eigenvectors make ML learning models easier to train because of the reduction of the information
- Examples of applications: in recommendation systems and financial risk analysis

Calculating Eigenvalues and Eigenvectors

- If we multiply matrix A with some vector v, it is the same as multiplying vector v by some scalar λ
- Vector v is called eigenvector and scalar lambda associated eigenvalue of matrix A

$$A \cdot \underset{\text{Eigenvector}}{\underbrace{v}} = \underset{\text{Eigenvalue of matrix A}}{\underbrace{\lambda}} \cdot v$$

$$A \cdot v - \lambda \cdot v = 0$$

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

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

$$A - \lambda \cdot I = 0$$

$$\begin{bmatrix} 3-\lambda & 4 \\ -1 & 7-\lambda \end{bmatrix} = 0$$

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{3} & \mathbf{4} \\ \mathbf{-1} & \mathbf{7} \end{bmatrix}$$

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

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

$$(\lambda - 5)^2 = 0$$

$$\boxed{\lambda_1 = \lambda_2 = 5}$$

$$\boxed{\lambda_1 = \lambda_2 = 5}$$

$$A - \lambda \cdot I = \begin{bmatrix} 3-\lambda & 4 \\ -1 & 7-\lambda \end{bmatrix} = \begin{bmatrix} 3-\mathbf{5} & 4 \\ -1 & 7-\mathbf{5} \end{bmatrix} = \begin{bmatrix} -2 & 4 \\ -1 & 2 \end{bmatrix} = \mathbf{0}$$

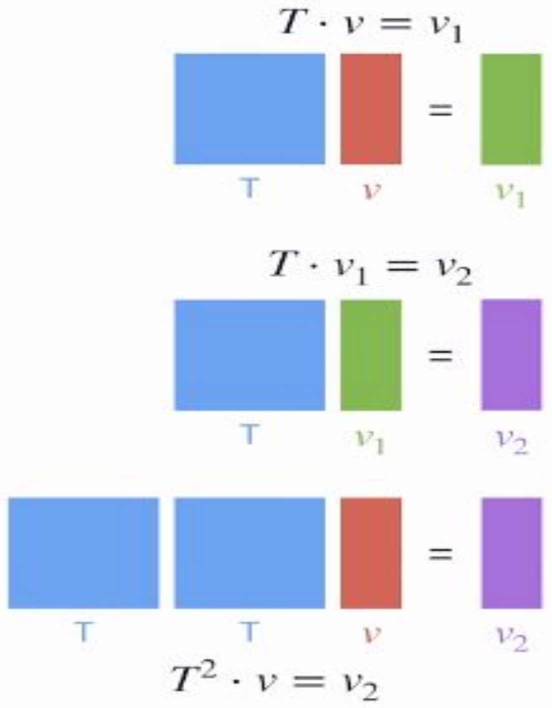
$$\begin{aligned} -2k_1 + 4k_2 &= 0 \\ -1k_1 + 2k_2 &= 0 \end{aligned} \longrightarrow \begin{aligned} k_1 &= 2k_2 \\ k_2 &= 2 \end{aligned} \longrightarrow k_1 = 2k_2 = 4$$

Changing to the Eigenbasis

- Calculate high powers of square matrix A
- Apply the same matrix multiplication many times
- Most efficient way to calculate A^n, especially for the larger values of *n*, is to first diagonalize A
- Diagonalizing a matrix involves finding its eigenvalues and eigenvectors

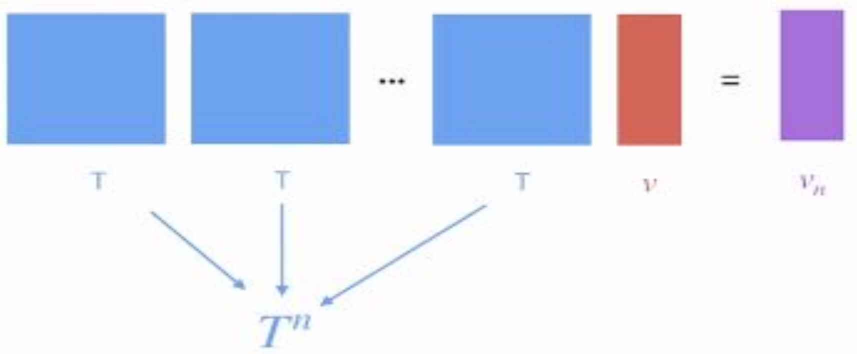
$$A^n = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \cdots \cdot \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

- Transformation matrix T represents rotation and shift of a vector v
- Result of applying the transformation T on a vector v by multiplying the T with v: we get a new vector v_1
- Applying the transformation T on the vector v_1 results in a new vector v_2
- This is equal to multiplying the transformation T twice with the vector v



- Calculate the position of the final vector v_n after n steps result of applying the transformation T on a vector v by multiplying the T with v, and we get a new vector v_1
- Calculate the position of the final vector v after one thousand steps
- Transform matrix T into a diagonal matrix; then this calculation would be easier and straightforward

$$v_n = T^n \cdot v$$



- Multiply a diagonal matrix by itself: multiply the diagonal elements by themselves, meaning you have to square each diagonal element
- To calculate the *n*th power of a diagonal matrix, we have to raise each of the elements on the diagonal to the power of *n*

$$\boldsymbol{D} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

$$D \cdot D = D^2 = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} = \begin{bmatrix} a^2 & 0 & 0 \\ 0 & a^2 & 0 \\ 0 & 0 & a^2 \end{bmatrix}$$

$$D^n = \begin{bmatrix} a^n & 0 & 0 \\ 0 & a^n & 0 \\ 0 & 0 & a^n \end{bmatrix}$$

- To create an eigenbasis conversion matrix, we have to plug in each of the eigenvectors as columns
- If *n* is a positive integer, λ is an eigenvalue of a matrix, and x is a corresponding eigenvector, then λ^n is an eigenvalue of T^n and x is a corresponding eigenvector

- Our problem of computing T^n can be simplified to the following: C is our eigenbasis conversion matrix, and D is a diagonalizable matrix
- To calculate matrix T^2, we have to multiply C D C^{-1} again with itself
- Formula for transform matrix T^n

$$T = C \cdot D \cdot C^{-1}$$

$$T^2 = C \cdot D \cdot \underbrace{C^{-1} C}_{I} \cdot D \cdot C^{-1}$$

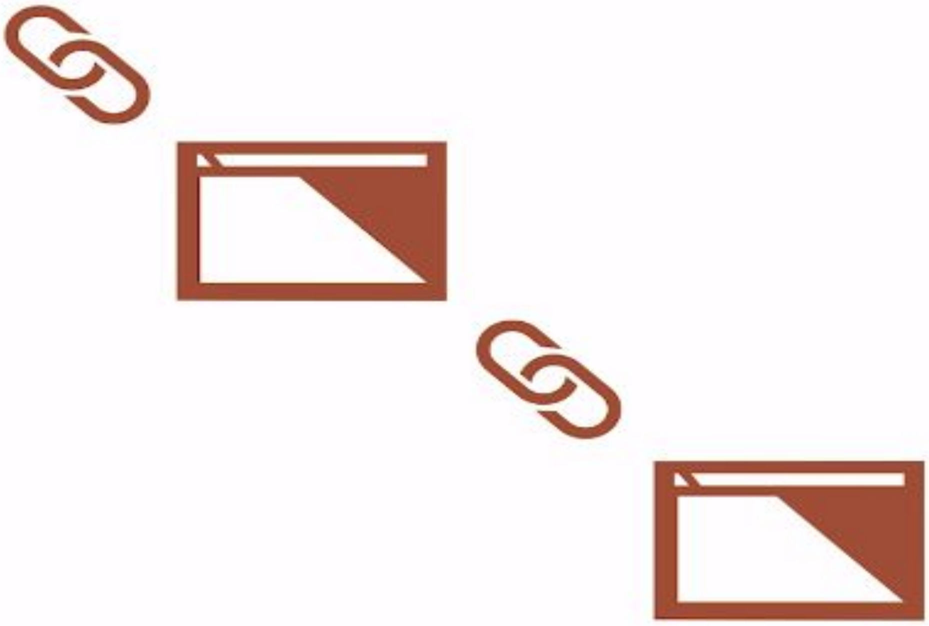
$$T^2 = C \cdot D \cdot D \cdot C^{-1}$$

$$T^2 = C \cdot D^2 \cdot C^{-1}$$

$$T^n = C \cdot D^n \cdot C^{-1}$$

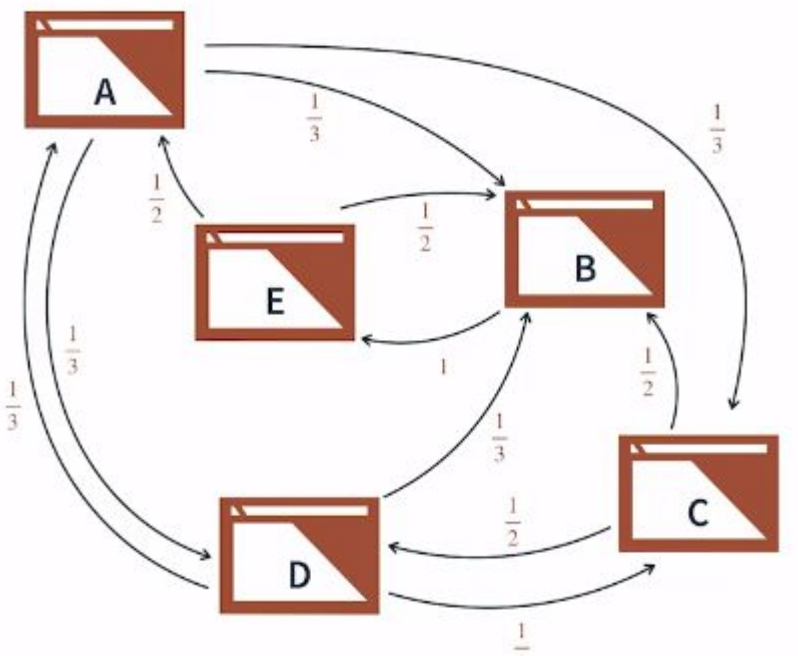
Google PageRank Algorithm

- Synonymous for link popularity, link value, link equity, and authority
- It counts the number and quality of the links to a page to determine how important a webpage is
- The more important a webpage is, the more likely it is to receive more links from other webpages



- Discover the best way to calculate the importance of each page that is returned by the query results
- To calculate a probability that can quantify the importance of a particular page
- Assigned a value, called a PageRank, to every page in its network of webpages

- We are analyzing five webpages in the network and name them A, B, C, D, and E
- Each webpage has links to other webpages
- Each link carries a fraction of the relevance that the webpage carries



- A network of links can be represented by a stochastic or probability matrix
- Each element is the probability of a link on a given webpage being selected and taking the user to another webpage
- Matrix A is connected to matrices B, C, and D, so we got $\frac{1}{3}$ for each of them, and the other two values in the first column are 0

$$M = \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \begin{bmatrix} 0 & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{2} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

- Denote by x1, x2, x3, x4, and x5 the importance of the five pages
- Find a solution of a system of five linear equations
- That can be done using eigenvalues and eigenvectors

$$x_1 = \frac{1}{3}x_4 + \frac{1}{2}x_5$$

$$x_2 = \frac{1}{3}x_1 + \frac{1}{2}x_3 + \frac{1}{3}x_4 + \frac{1}{2}x_5$$

$$x_3 = \frac{1}{3}x_1 + \frac{1}{3}x_4$$

$$x_4 = \frac{1}{3}x_1 + \frac{1}{2}x_3$$

$$x_5 = 1x_1$$


Risk Matrix
Med
High
Critical
Critical
Low
Med
High
Critical
Low
Low
Med
High
Low
Low
Low
Med
Likelihood
Impact