# Линейная алгебра, тема 3. Матрицы

## Линейная (не)зависимость

Если линейная комбинация $k_1 \cdot \vec{a_1} + k_2 \cdot \vec{a_2} + ... + k_n \cdot \vec{a_n}$ равна нулевому вектору только при $k_1 = k_2 = ... = k_n = 0$, то система векторов $\vec{a_1}$, $\vec{a_2}$, ..., $\vec{a_n}$ называется **линейно независимой**.

Во всех остальных случаях система векторов **линейно зависимая**: для этого достаточно хотя бы одного скаляра, не равного нулю.

Система из одного вектора линейно зависима только тогда, когда этот вектор нулевой.

**Проверка системы на линейную (не)зависимость**

1. Через пропорциональность координат.

   Если система векторов содержит векторы $\vec{a}$ и $k \cdot \vec{a}$, где $k$ — произвольное число, отличное от нуля, то она линейно зависима.

   Для проверки достаточно найти коэффициент коллинеарности векторов: если на всех парах координат он совпадает, векторы коллинеарны и система линейно зависима.

   Возьмём, например векторы $\vec{a} = (2, -4)$, $\vec{b} = (-3, -6)$, $\vec{c} = (1, 2)$.

   Система векторов $\vec{a}$, $\vec{c}$ линейно независима, так как $\frac{2}{1} \neq \frac{-4}{2}$.

   Система векторов $\vec{b}$, $\vec{c}$ линейно зависима, так как $\frac{-3}{1} = \frac{-6}{2}$.

2. Через количество векторов и координат

   В векторном пространстве размерности $n$ система из $k$ векторов:

   - линейно независима, если $k < n$ и все векторы ненулевые;
   - может быть как линейно зависимой, так и линейно независимой, если $k = n$;

- линейно зависима, если $k > n$.

2. Через сумму векторов

   Если один из векторов в пространстве является суммой двух других, то система из этих трёх векторов будет линейно зависимой.

## Базис

### Нормированное пространство

Число, соответствующее максимальному количеству линейно независимых векторов в пространстве, называют **размерностью векторного пространства**. Это число совпадает с количеством координат вектора.

**Единичный вектор** (или **орт**) — это вектор нормированного пространства, длина которого равна $1$.

Процесс преобразования вектора $\vec{a}$ в единичный вектор $\vec{u}$ называют нормированием. Чтобы нормировать вектор, нужно разделить его на его длину.

$$\vec{u} = \frac{\vec{a}}{|\vec{a}|}$$

### Базис

**Базис векторного пространства** — упорядоченная система линейно независимых векторов, число векторов в которой равно размерности векторного пространства. Например, у трёхмерного пространства базис будет состоять из трёх векторов, у каждого из которых по $3$ координаты.

Любой вектор $\vec{a}$ на плоскости можно единственным образом записать как линейную комбинацию базисных векторов:
$\vec{a} = k_1 \cdot \vec{i} + k_2 \cdot \vec{j},$

где $k_1$, $k_2$ — числа, которые называют **координатами** вектора в данном базисе.

А само выражение $\vec{a} = k_1 \cdot \vec{i} + k_2 \cdot \vec{j}$ называют **разложением вектора $\vec{a}$ по базису $\{\vec{i}, \vec{j}\}$.**

Ортонормированный базис плоскости обычно обозначают $\{\vec{i}, \vec{j}\}$,
ортонормированный базис пространства — $\{\vec{i}, \vec{j}, \vec{k}\}$.

**Матрица** — прямоугольная таблица чисел. Эти числа называют **элементами матрицы.**

**Размер (порядок) матрицы** — количество строк и столбцов. Сначала пишут количество строк, потом столбцов.

В общем виде матрицу записывают так:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ ... & ... & ... & ... & ... \\ a_{m1} & a_{m2} & a_{m3} & ... & a_{mn} \end{pmatrix} = (a_{ij})_{m \times n}$$

Это матрица размера $m \times n$, её элемент $a_{ij}$ расположен на пересечении $i$-той строки и $j$-того столбца. Запись $A = (a_{ij})_{m \times n}$ определяет имя элементов матрицы и её размеры.

Если поменять местами строки или столбцы, получится матрица, **эквивалентная** данной. Перестановкой отдельных элементов эквивалентную матрицу не получить.

**Транспонировать** матрицу $A = (a_{ij})_{m \times n}$ значит, записать её строки в столбцы с сохранением порядка следования. В результате получается матрица $A^T = (a_{ji})_{n \times m}$.
Транспонированную матрицу помечают надстрочным индексом $^T$.

## Умножение на скаляр и сложение матриц

Чтобы умножить или разделить матрицу на число, нужно умножить или разделить на него каждый элемент этой матрицы.

Складывать можно только матрицы одного размера. Чтобы найти сумму $A = (a_{ij})_{m \times n}$ и $B = (b_{ij})_{m \times n}$, элемент $a_{11}$ первой матрицы нужно сложить с элементом $b_{11}$ второй матрицы, элемент $a_{34}$ первой матрицы — с элементом $b_{34}$ второй матрицы и так далее.

Точно так же происходит вычитание матриц.

**Свойства умножения**

1) $1 \cdot A = A$;

2) $-1 \cdot A = -A$;

3) $0 \cdot A = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$. Это нулевая матрица. Она будет тех же размеров, что и матрица $A$;

4) Для любого числа $k \in \mathbb{R}$ $k \cdot A = A \cdot k$ — умножение на скаляр коммутативно;

5) Для любых чисел $k, m \in \mathbb{R}$ $k \cdot (m \cdot A) = (k \cdot m) \cdot A$ — можно сначала найти произведение скаляров, а потом умножать на матрицу;

6) $(k \cdot A)^T = k \cdot A^T$ — сначала транспонируем, потом умножаем на скаляр.

**Свойства сложения**

Для произвольных матриц $A = (a_{ij})_{m \times n}$, $B = (b_{ij})_{m \times n}$, $C = (c_{ij})_{m \times n}$ одинакового размера $m \times n$:

1) $A + B = B + A$ — свойство коммутативности;

2) $(A + B) + C = A + (B + C)$ — свойство ассоциативности;

3) $A + O = A$ — прибавление нулевой матрицы;

4) $\forall\ A\ \exists!\ B$: $A + B = O$ — для любой матрицы существует единственная, противоположная ей;

5) $(A + B)^T = A^T + B^T$ — транспонированная сумма матриц равна сумме этих транспонированных матриц.

**Свойства дистрибутивности**

Для произвольных матриц $A = (a_{ij})_{m \times n}$, $B = (b_{ij})_{m \times n}$ одинакового размера $m \times n$ и чисел $k, n \in \mathbb{R}$

1) $(k + n) \cdot A = k \cdot A + n \cdot A$;

2) $k \cdot (A + B) = k \cdot A + k \cdot B$.

**Порядок действий**

1) Транспонирование;

2) Умножение на скаляр;

3) Сложение матриц.

## Умножение матрицы на вектор

Умножение вектора и матрицы происходит по схеме «строка на столбец»: итоговый вектор состоит из скалярных произведений строк первого множителя на столбцы второго множителя. Такое умножение не коммутативно.

$k_{1\times m}\cdot M_{m\times n}=(k\cdot M)_{1\times n}$

$M_{m\times n}\cdot k_{n\times 1}=(M\cdot k)_{m\times 1}$

По умолчанию используют векторы-столбцы.

Скалярное произведение векторов в матричном виде:

$\vec{a}\cdot\vec{b}=a^T\cdot b.$

Отображение $A:V\to V$ называют **линейным преобразованием** векторного пространства, если для любых векторов $\vec{x}$, $\vec{y}$ этого пространства и числа $n$:
1) $A(\vec{x}+\vec{y})=A(\vec{x})+A(\vec{y})$;
2) $A(n\cdot\vec{x})=n\cdot A(\vec{x})$.

**Растяжение/сжатие**: $\begin{pmatrix}a & 0\\ 0 & b\end{pmatrix}$.

**Сдвиг**: $\begin{pmatrix}1 & a\\ 0 & 1\end{pmatrix}$ и $\begin{pmatrix}1 & 0\\ b & 1\end{pmatrix}$.

**Вращение**: $\begin{pmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{pmatrix}$.

Матрица $\begin{pmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{pmatrix}$ позволит записать координаты вектора плоскости $xOy$ в пространстве.

Аналогично матрицы $\begin{pmatrix}1 & 0\\ 0 & 0\\ 0 & 1\end{pmatrix}$, $\begin{pmatrix}0 & 0\\ 1 & 0\\ 0 & 1\end{pmatrix}$ дадут координаты векторов в плоскостях $xOz$ и $yOz$.

Если заменить матрицы на транспонированные $\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}$, $\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}$, $\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$, то можно построить проекции вектора на плоскостях $xOy$, $xOz$, $yOz$.

## Матричные перемножения

Для умножения матрицы $A$ на матрицу $B$ количество столбцов $A$ должно быть равно количеству строк $B$.

$A_{m\times n} \cdot B_{n\times k} = C_{m\times k}$. Часто произведение $C$ записывают просто как $AB$.

При умножении $m$-ной строки первой матрицы на $k$-тый столбец второй матрицы получается элемент $c_{mk}$ произведения.

Матрицы, для которых $AB = BA$, называют **перестановочными** или коммутативными**.**

**Единичной матрицей** $E$ называют диагональную матрицу, все элементы на главной диагонали которой равны $1$.

**Свойства умножения**

1) $AB \neq BA$, если они не перестановочные;

2) $A(BC) = (AB)C$ — ассоциативность произведения;

3) $AE = EA = A$;

4) Дистрибутивность относительно сложения:

$(A + B) \cdot C = AC + BC$;

$K(A + B) = KA + KB$;

5) Ассоциативность умножения на скаляр:

$(\alpha AB) = \alpha(AB) = A(\alpha B)$;

6) Транспонирование произведения:

$(AB)^T = B^T \cdot A^T$;

$(kA)^T = A^T \cdot k^T$;

$(Am)^T = m^T \cdot A^T$.

Чтобы применить линейное преобразование к нескольким векторам сразу, их можно объединить в одну матрицу и умножить её на матрицу преобразования.

## Операции с матрицами в Python

В коде матрицу записывают построчно, помещая каждую строку в свои квадратные скобки. Удобнее писать каждую строчку с новой строки. При этом все

они будут расположены друг под другом внутри функции `array()`.

Чтобы транспонировать матрицу `m`, достаточно записать операцию `m.T`.

```
import numpy as np

m = np.array([
    [2, 17, 43],
    [36, 8, 4],
    [10, -11, 6],
    [3, 7, 15]
  ])
print("Матрица:")
print(m)  # Вывод исходной матрицы.
print()  # Пустая строка для разделения.
m = m.T  # Теперь m - это транспонированная матрица.
print("Транспонированная матрица:")
print(m)  # Вывод новой, транспонированной матрицы.
```

В коде нумерация строк и столбцов идёт с нуля.

Чтобы вывести на экран конкретный элемент, нужно записать имя матрицы и номер элемента в квадратных скобках.

Чтобы вывести целую строку, второй параметр нужно заменить на `:`.
Чтобы вывести целый столбец, первый параметр нужно заменить на `:`.
В данном случае это означает «взять все значения».

```
# Вывод элемента в 4 строке, 1 столбце.
print(friends[4, 1])
# Вывод 5 строки.
print(friends[5, :])
```

**Сложение** матриц в Python выглядит так же, как сложение векторов: задаём матрицы и находим их сумму с помощью знака `+`. Знак **умножения на скаляр**: `*`.

**Умножение матрицы на вектор**. Пример:

```
import numpy as np

a = np.array([
    [5, -7, 2],
    [-2, -1, 9],
    [3, 1, 4]
```

```
])
k = np.array([3, 4, 5])
b = np.array([
    [0, 7, 1.2],
    [4, -5, -3],
    [3, 0.1, 12]
])
a.shape # Размер матрицы a.
k.shape # Размер вектора k.
k_column = k.reshape(3, 1) # Запись вектора матрицей 3x1.
k_string = k.reshape(1, 3) # Запись вектора матрицей 1x3.
print("Произведение вектора на матрицу:")
print(k_string @ a)
print()
print("Произведение матрицы на вектор:")
print(a @ k_column)
print("Произведение AB:")
print(a @ b)
print("Произведение BA:")
print(b @ a)
```