



QR-разложение и разложение Шура.

Даня Меркулов

МФТИ. AI360

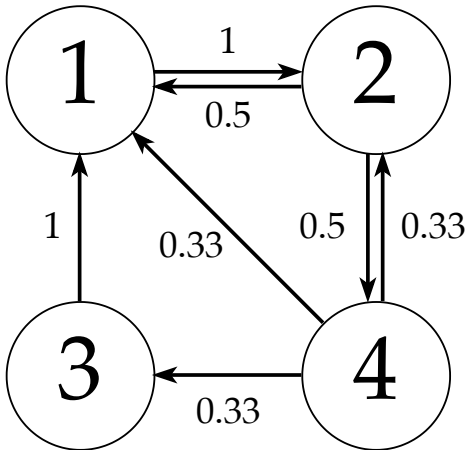
# PageRank

## PageRank

Рассмотрим простой пример. Предположим, что у нас есть 4 веб-сайта с некоторыми ссылками. Наша цель --- понять, насколько важен каждый из этих сайтов. Очевидно, что мы можем переформулировать эту проблему в терминах ориентированных графов. Здесь каждый узел представляет веб-сайт, а каждое ребро описывает ссылку с одного сайта на другой.



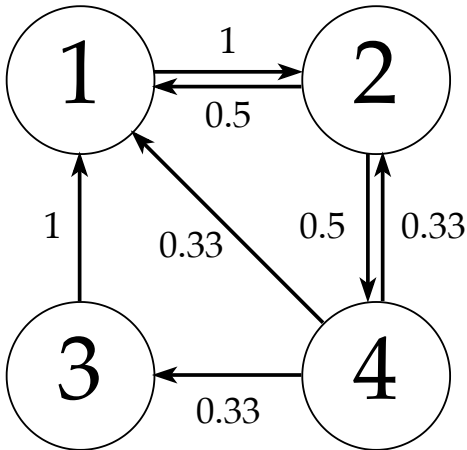
## PageRank



Таким образом, мы можем ввести матрицу переходов  $A$ :

$$A = \begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ 1 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

## PageRank



Таким образом, мы можем ввести матрицу переходов  $A$ :

$$A = \begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ 1 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

Давайте введём вектор PageRank  $x$ , который описывает важность каждого веб-сайта.

$x = (x_1, x_2, x_3, x_4)^T$ , где  $x_i$  - важность  $i$ -го веб-сайта

## PageRank



Таким образом, мы можем ввести матрицу переходов  $A$ :

$$A = \begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ 1 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

Давайте введём вектор PageRank  $x$ , который описывает важность каждого веб-сайта.

$x = (x_1, x_2, x_3, x_4)^T$ , где  $x_i$  - важность  $i$ -го веб-сайта

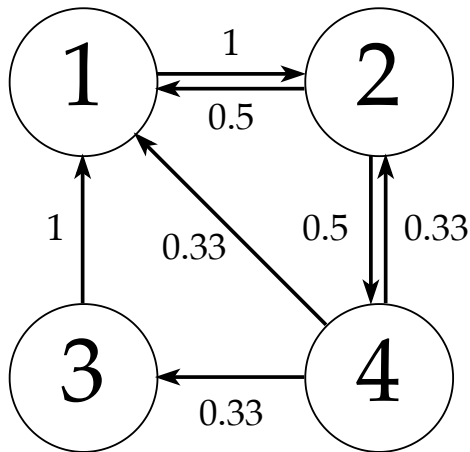
Предположим, что начальная важность равномерно распределена между всеми узлами. Тогда:

$$x^0 = (0.25, 0.25, 0.25, 0.25)^T$$

Каждая входящая ссылка увеличивает важность узла. Таким образом, это обновление может быть записано как умножение матрицы на вектор:

$$x^1 = A \cdot x^0 = (0.46, 0.33, 0.08, 0.125)^T$$

## PageRank



Повторяя те же операции, мы можем легко увидеть сходимость:

$$\mathbf{x}^2 = A \cdot \mathbf{x}^1 = (0.29, 0.50, 0.04, 0.17)^\top$$

$$\mathbf{x}^3 = A \cdot \mathbf{x}^2 = (0.35, 0.35, 0.06, 0.25)^\top$$

...

$$\mathbf{x}^{14} = A \cdot \mathbf{x}^{13} = (0.33, 0.40, 0.07, 0.20)^\top$$

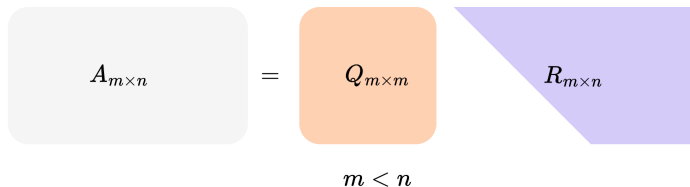
$$\mathbf{x}^{15} = A \cdot \mathbf{x}^{14} = (0.33, 0.40, 0.07, 0.20)^\top$$

Выполните упражнение ♣️PageRank.

## QR-разложение



## QR-разложение матрицы



Для произвольной матрицы  $A \in \mathbb{R}^{m \times n}$  существует разложение вида:

$$A = QR,$$

где  $Q \in \mathbb{R}^{m \times m}$  - ортогональная матрица ( $Q^T Q = Q Q^T = I$ ), а  $R \in \mathbb{R}^{m \times n}$  - верхнетреугольная матрица. В случае, когда  $m > n$ , матрица  $Q$  может быть усечена до размера  $m \times n$  с сохранением ортогональности столбцов. Если зафиксировать все диагональные элементы матрицы  $R$ , то разложение становится единственным.

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



Рис. 1: Иллюстрация процесса Грама-Шмидта

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .

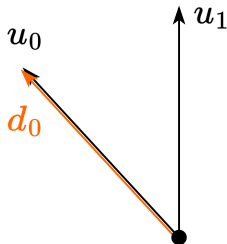


Рис. 2: Иллюстрация процесса Грама-Шмидта

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



Рис. 3: Иллюстрация процесса Грама-Шмидта

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



Рис. 4: Иллюстрация процесса Грама-Шмидта

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

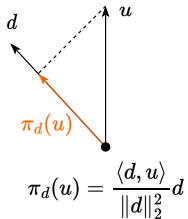
**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



Рис. 5: Иллюстрация процесса Грама-Шмидта

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .



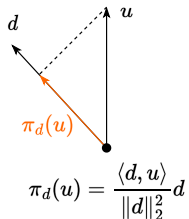
# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



$$d_0 = u_0$$



$$\pi_d(u) = \frac{\langle d, u \rangle}{\|d\|_2^2} d$$



# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



$$d_0 = u_0$$

$$d_1 = u_1 - \pi_{d_0}(u_1)$$



# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

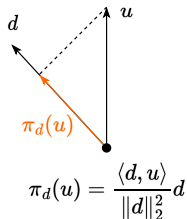
**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



$$d_0 = u_0$$

$$d_1 = u_1 - \pi_{d_0}(u_1)$$

$$d_2 = u_2 - \pi_{d_0}(u_2) - \pi_{d_1}(u_2)$$



$$\pi_d(u) = \frac{\langle d, u \rangle}{\|d\|_2^2} d$$

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .

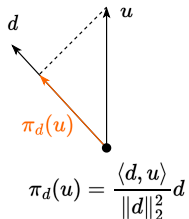


$$d_0 = u_0$$

$$d_1 = u_1 - \pi_{d_0}(u_1)$$

$$d_2 = u_2 - \pi_{d_0}(u_2) - \pi_{d_1}(u_2)$$

$\vdots$

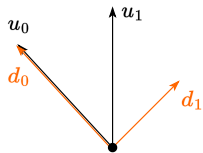


$$\pi_d(u) = \frac{\langle d, u \rangle}{\|d\|_2^2} d$$

# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



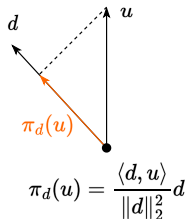
$$d_0 = u_0$$

$$d_1 = u_1 - \pi_{d_0}(u_1)$$

$$d_2 = u_2 - \pi_{d_0}(u_2) - \pi_{d_1}(u_2)$$

$\vdots$

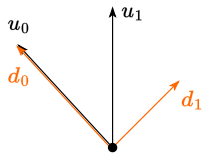
$$d_k = u_k - \sum_{i=0}^{k-1} \pi_{d_i}(u_k)$$



# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



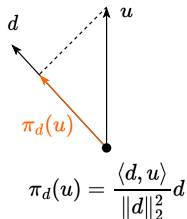
$$d_0 = u_0$$

$$d_1 = u_1 - \pi_{d_0}(u_1)$$

$$d_2 = u_2 - \pi_{d_0}(u_2) - \pi_{d_1}(u_2)$$

$\vdots$

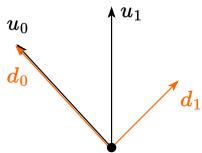
$$d_k = u_k - \sum_{i=0}^{k-1} \pi_{d_i}(u_k)$$



# Процесс Грама-Шмидта

**Вход:**  $n$  линейно независимых векторов  $u_0, \dots, u_{n-1}$ .

**Выход:**  $n$  линейно независимых попарно ортогональных векторов  $d_0, \dots, d_{n-1}$ .



$$d_0 = u_0$$

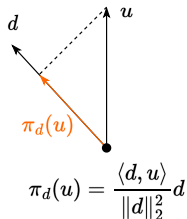
$$d_1 = u_1 - \pi_{d_0}(u_1)$$

$$d_2 = u_2 - \pi_{d_0}(u_2) - \pi_{d_1}(u_2)$$

$\vdots$

$$d_k = u_k - \sum_{i=0}^{k-1} \pi_{d_i}(u_k)$$

$$d_k = u_k + \sum_{i=0}^{k-1} \beta_{ik} d_i \quad \beta_{ik} = -\frac{\langle d_i, u_k \rangle}{\langle d_i, d_i \rangle} \quad (1)$$



# QR-разложение с помощью процесса Грама-Шмидта

Пусть дана матрица  $A \in \mathbb{R}^{m \times n}$ , у которой столбцы - это векторы

$$A = [u_1 \ u_2 \ \dots \ u_n].$$

Если мы последовательно применим процесс Грама-Шмидта к этим столбцам:

1. На шаге  $k$  мы удаляем из столбца  $u_k$  проекции на все ранее полученные ортогональные векторы  $d_1, \dots, d_{k-1}$ .

# QR-разложение с помощью процесса Грама-Шмидта

Пусть дана матрица  $A \in \mathbb{R}^{m \times n}$ , у которой столбцы - это векторы

$$A = [u_1 \ u_2 \ \dots \ u_n].$$

Если мы последовательно применим процесс Грама-Шмидта к этим столбцам:

1. На шаге  $k$  мы удаляем из столбца  $u_k$  проекции на все ранее полученные ортогональные векторы  $d_1, \dots, d_{k-1}$ .
2. Нормируем результат и получаем ортонормированный вектор  $q_k = \frac{d_k}{\|d_k\|}$ .



# QR-разложение с помощью процесса Грама-Шмидта

Пусть дана матрица  $A \in \mathbb{R}^{m \times n}$ , у которой столбцы - это векторы

$$A = [u_1 \ u_2 \ \dots \ u_n].$$

Если мы последовательно применим процесс Грама-Шмидта к этим столбцам:

1. На шаге  $k$  мы удаляем из столбца  $u_k$  проекции на все ранее полученные ортогональные векторы  $d_1, \dots, d_{k-1}$ .
2. Нормируем результат и получаем ортонормированный вектор  $q_k = \frac{d_k}{\|d_k\|}$ .

# QR-разложение с помощью процесса Грама-Шмидта

Пусть дана матрица  $A \in \mathbb{R}^{m \times n}$ , у которой столбцы - это векторы

$$A = [u_1 \ u_2 \ \dots \ u_n].$$

Если мы последовательно применим процесс Грама-Шмидта к этим столбцам:

1. На шаге  $k$  мы удаляем из столбца  $u_k$  проекции на все ранее полученные ортогональные векторы  $d_1, \dots, d_{k-1}$ .
2. Нормируем результат и получаем ортонормированный вектор  $q_k = \frac{d_k}{\|d_k\|}$ .

Таким образом образуется ортонормированный набор столбцов

$$Q = [q_1 \ q_2 \ \dots \ q_n].$$

А коэффициенты, которые появляются при разложении каждого  $u_k$  по векторам  $q_1, \dots, q_k$ , образуют верхнетреугольную матрицу  $R$ :

$$u_k = \underbrace{\langle q_1, u_k \rangle}_{r_{1k}} q_1 + \underbrace{\langle q_2, u_k \rangle}_{r_{2k}} q_2 + \dots + \underbrace{\langle q_k, u_k \rangle}_{r_{kk}} q_k.$$

Все элементы  $r_{ij} = 0$  при  $j < i$ , и в итоге получаем **QR-разложение**:

$$A = QR,$$

где  $Q$  - ортонормированная (ортогональная) матрица, а  $R$  - верхняя треугольная.

## Модификация процесса Грама-Шмидта

- Процесс Грама-Шмидта может быть **численно неустойчив**, то есть векторы могут перестать быть ортогональными в машинной арифметике, особенно если норма  $q_k$  мала. Это явление называется **потерей ортогональности**.

# Модификация процесса Грама-Шмидта

- Процесс Грама-Шмидта может быть **численно неустойчив**, то есть векторы могут перестать быть ортогональными в машинной арифметике, особенно если норма  $q_k$  мала. Это явление называется **потерей ортогональности**.
- Существует метод **modified Gram-Schmidt** (MGS). Вместо того, чтобы сразу вычитать все проекции:

$$q_k := a_k - (a_k, q_1)q_1 - \dots - (a_k, q_{k-1})q_{k-1},$$

мы делаем это **поэтапно**:

$$q_k := a_k,$$

$$q_k := q_k - (q_k, q_1) q_1,$$

$$q_k := q_k - (q_k, q_2) q_2,$$

$$\vdots$$

# Модификация процесса Грама-Шмидта

- Процесс Грама-Шмидта может быть **численно неустойчив**, то есть векторы могут перестать быть ортогональными в машинной арифметике, особенно если норма  $q_k$  мала. Это явление называется **потерей ортогональности**.
- Существует метод **modified Gram-Schmidt** (MGS). Вместо того, чтобы сразу вычитать все проекции:

$$q_k := a_k - (a_k, q_1)q_1 - \dots - (a_k, q_{k-1})q_{k-1},$$

мы делаем это **поэтапно**:

$$q_k := a_k,$$

$$q_k := q_k - (q_k, q_1)q_1,$$

$$q_k := q_k - (q_k, q_2)q_2,$$

$$\vdots$$

- В точной арифметике результат совпадает со стандартной процедурой Грама-Шмидта, но в машинной арифметике это даёт **совершенно другой** (намного более устойчивый) результат.

# Модификация процесса Грама-Шмидта

- Процесс Грама-Шмидта может быть **численно неустойчив**, то есть векторы могут перестать быть ортогональными в машинной арифметике, особенно если норма  $q_k$  мала. Это явление называется **потерей ортогональности**.
- Существует метод **modified Gram-Schmidt** (MGS). Вместо того, чтобы сразу вычитать все проекции:

$$q_k := a_k - (a_k, q_1)q_1 - \dots - (a_k, q_{k-1})q_{k-1},$$

мы делаем это **поэтапно**:

$$q_k := a_k,$$

$$q_k := q_k - (q_k, q_1) q_1,$$

$$q_k := q_k - (q_k, q_2) q_2,$$

$$\vdots$$

- В точной арифметике результат совпадает со стандартной процедурой Грама-Шмидта, но в машинной арифметике это даёт **совершенно другой** (намного более устойчивый) результат.
- Сложность модифицированного процесса Грама-Шмидта составляет  $\mathcal{O}(n^2 m)$  операций.

## Упражнение

Выполните упражнение  Модифицированный процесс Грама-Шмидта.

## Алгоритм Хаусхолдера для построения QR-разложения

- Используя полученное свойство мы можем сделать произвольную матрицу  $A$  нижней треугольной:

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$



## Алгоритм Хаусхолдера для построения QR-разложения

- Используя полученное свойство мы можем сделать произвольную матрицу  $A$  нижней треугольной:

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

- Затем находим  $H_3 = \begin{bmatrix} I_2 & \\ & \tilde{H}_3 \end{bmatrix}$  такую, что

$$\tilde{H}_3 \begin{bmatrix} \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \end{bmatrix}.$$

## Алгоритм Хаусхолдера для построения QR-разложения

- Используя полученное свойство мы можем сделать произвольную матрицу  $A$  нижней треугольной:
- Получаем

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

$$H_3 H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

- Затем находим  $H_3 = \begin{bmatrix} I_2 & \\ & \tilde{H}_3 \end{bmatrix}$  такую, что

$$\tilde{H}_3 \begin{bmatrix} \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \end{bmatrix}.$$

## Алгоритм Хаусхолдера для построения QR-разложения

- Используя полученное свойство мы можем сделать произвольную матрицу  $A$  нижней треугольной:
- Получаем

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

- Затем находим  $H_3 = \begin{bmatrix} I_2 & \\ & \tilde{H}_3 \end{bmatrix}$  такую, что

$$\tilde{H}_3 \begin{bmatrix} \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \end{bmatrix}.$$

$$H_3 H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

- Аналогично находим  $H_4$  и получаем верхнюю треугольную матрицу.

## Алгоритм Хаусхолдера для построения QR-разложения

- Используя полученное свойство мы можем сделать произвольную матрицу  $A$  нижней треугольной:
- Получаем

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

- Затем находим  $H_3 = \begin{bmatrix} I_2 & \\ & \tilde{H}_3 \end{bmatrix}$  такую, что

$$\tilde{H}_3 \begin{bmatrix} \times \\ \times \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \end{bmatrix}.$$

$$H_3 H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

- Аналогично находим  $H_4$  и получаем верхнюю треугольную матрицу.
- Так как произведение ортогональных и обратная к ортогональной матрицы являются ортогональными матрицами, мы получаем **следствие:** (QR-разложение) Любая  $A \in \mathbb{R}^{n \times m}$  может быть представлена как

$$A = QR,$$

где  $Q$  - ортогональная и  $R$  - верхняя треугольная. См. постер, каковы размеры  $Q$  и  $R$  для  $n > m$  и  $n < m$ .

## Матрица вращения Гивенса (Якоби)

- Матрица вращения Гивенса (Якоби) - это ортогональная матрица, которая используется для вращения вектора в плоскости на угол  $\alpha$ . Она имеет следующий вид:

$$G = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$$

Легко проверить, что  $G^T G = G G^T = I$ , то есть матрица является ортогональной.

## Матрица вращения Гивенса (Якоби)

- Матрица вращения Гивенса (Якоби) - это ортогональная матрица, которая используется для вращения вектора в плоскости на угол  $\alpha$ . Она имеет следующий вид:

$$G = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$$

Легко проверить, что  $G^T G = G G^T = I$ , то есть матрица является ортогональной.

- Для общего случая размерности  $n$  мы выбираем две координаты  $(i, j)$  и выполняем вращение вектора  $x$  только в этой плоскости:

$$x' = Gx,$$

где изменяются только  $i$ -я и  $j$ -я координаты:

$$x'_i = x_i \cos \alpha - x_j \sin \alpha, \quad x'_j = x_i \sin \alpha + x_j \cos \alpha,$$

при этом остальные  $x_k$  остаются неизменными.

## Матрица вращения Гивенса (Якоби)

- Матрица вращения Гивенса (Якоби) - это ортогональная матрица, которая используется для вращения вектора в плоскости на угол  $\alpha$ . Она имеет следующий вид:

$$G = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$$

Легко проверить, что  $G^T G = G G^T = I$ , то есть матрица является ортогональной.

- Для общего случая размерности  $n$  мы выбираем две координаты  $(i, j)$  и выполняем вращение вектора  $x$  только в этой плоскости:

$$x' = Gx,$$

где изменяются только  $i$ -я и  $j$ -я координаты:

$$x'_i = x_i \cos \alpha - x_j \sin \alpha, \quad x'_j = x_i \sin \alpha + x_j \cos \alpha,$$

при этом остальные  $x_k$  остаются неизменными.

- Чтобы обнулить  $j$ -ю координату вектора, выбираем угол  $\alpha$  так, что:

$$\cos \alpha = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad \sin \alpha = -\frac{x_j}{\sqrt{x_i^2 + x_j^2}}$$

## Матрица вращения Гивенса (Якоби)

- Матрица вращения Гивенса (Якоби) - это ортогональная матрица, которая используется для вращения вектора в плоскости на угол  $\alpha$ . Она имеет следующий вид:

$$G = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$$

Легко проверить, что  $G^T G = G G^T = I$ , то есть матрица является ортогональной.

- Для общего случая размерности  $n$  мы выбираем две координаты  $(i, j)$  и выполняем вращение вектора  $x$  только в этой плоскости:

$$x' = Gx,$$

где изменяются только  $i$ -я и  $j$ -я координаты:

$$x'_i = x_i \cos \alpha - x_j \sin \alpha, \quad x'_j = x_i \sin \alpha + x_j \cos \alpha,$$

при этом остальные  $x_k$  остаются неизменными.

- Чтобы обнулить  $j$ -ю координату вектора, выбираем угол  $\alpha$  так, что:

$$\cos \alpha = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad \sin \alpha = -\frac{x_j}{\sqrt{x_i^2 + x_j^2}}$$

- Применяя последовательно матрицы Гивенса, можно привести матрицу к верхнетреугольному виду: для этого нужно  $n - 1$  вращений, чтобы обнулить элементы под главной диагональю в каждом столбце.



## QR через вращения Гивенса

Также мы можем сделать матрицу верхнетреугольной с помощью вращений Гивенса:

$$\begin{bmatrix} \times & \times & \times \\ * & \times & \times \\ * & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} * & \times & \times \\ * & \times & \times \\ 0 & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times \\ 0 & * & \times \\ 0 & * & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

## Вращения Гивенса vs. отражения Хаусхолдера

- Отражения Хаусхолдера полезны для плотных матриц (сложность приблизительно в два раза меньше, чем для Якоби) и мы должны обнулить большое количество элементов.

## Вращения Гивенса vs. отражения Хаусхолдера

- Отражения Хаусхолдера полезны для плотных матриц (сложность приблизительно в два раза меньше, чем для Якоби) и мы должны обнулить большое количество элементов.
- Вращения Гивенса более подходят для разреженных матриц или параллельных вычислений, так как они действуют локально на элементах.

## Упражнение: QR-разложение вектора $(1, 2)$

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

## Упражнение: QR-разложение вектора (1, 2)

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

**Решение:**

1. Вычисляем норму вектора:

$$\|a\| = \sqrt{1^2 + 2^2} = \sqrt{5}.$$

## Упражнение: QR-разложение вектора (1, 2)

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

**Решение:**

1. Вычисляем норму вектора:

$$\|a\| = \sqrt{1^2 + 2^2} = \sqrt{5}.$$

2. Получаем ортонормированный вектор:

$$q_1 = \frac{a}{\|a\|} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

## Упражнение: QR-разложение вектора (1, 2)

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

**Решение:**

1. Вычисляем норму вектора:

$$\|a\| = \sqrt{1^2 + 2^2} = \sqrt{5}.$$

2. Получаем ортонормированный вектор:

$$q_1 = \frac{a}{\|a\|} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

3. Поскольку вектор один, то QR-разложение имеет вид:

$$a = Q \cdot R,$$

где

$$Q = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad R = \sqrt{5}.$$

## Упражнение: QR-разложение вектора $(1, 2, 3)$

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$



## Упражнение: QR-разложение вектора $(1, 2, 3)$

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

**Решение:**

1. Вычисляем норму вектора:

$$\|a\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}.$$

## Упражнение: QR-разложение вектора $(1, 2, 3)$

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

**Решение:**

1. Вычисляем норму вектора:

$$\|a\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}.$$

2. Получаем ортонормированный вектор:

$$q_1 = \frac{a}{\|a\|} = \frac{1}{\sqrt{14}} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

## Упражнение: QR-разложение вектора (1, 2, 3)

Рассмотрим вектор

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

**Решение:**

1. Вычисляем норму вектора:

$$\|a\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}.$$

2. Получаем ортонормированный вектор:

$$q_1 = \frac{a}{\|a\|} = \frac{1}{\sqrt{14}} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

3. Так как вектор один, то имеем:

$$a = Q \cdot R,$$

где

$$Q = \frac{1}{\sqrt{14}} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad R = \sqrt{14}.$$

## Разложение Шура

# Разложение Шура

Для произвольной квадратной матрицы  $A \in \mathbb{R}^{n \times n}$  существует разложение:

$$A = UTU^*,$$

где  $U$  - унитарная матрица,  $T$  - верхняя треугольная матрица с собственными числами матрицы  $A$  на главной диагонали.

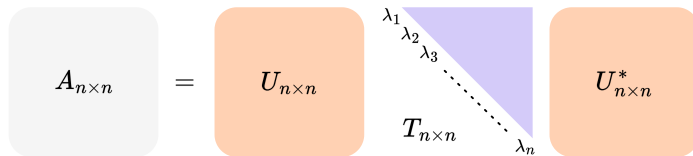


Рис. 6: Разложение Шура матрицы  $A$

## QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$

# QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :

# QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :
  - Вычисляем QR-разложение:  $A_k = Q_k R_k$



# QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :
  - Вычисляем QR-разложение:  $A_k = Q_k R_k$
  - Формируем следующую итерацию:  $A_{k+1} = R_k Q_k$

# QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :
  - Вычисляем QR-разложение:  $A_k = Q_k R_k$
  - Формируем следующую итерацию:  $A_{k+1} = R_k Q_k$

## QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :
  - Вычисляем QR-разложение:  $A_k = Q_k R_k$
  - Формируем следующую итерацию:  $A_{k+1} = R_k Q_k$

Для симметричных матриц этот процесс сходится к диагональной матрице, содержащей собственные числа. Для произвольных матриц он сходится к верхнетреугольной матрице, где диагональные элементы являются собственными числами исходной матрицы.

$$A_{k+1} = R_k Q_k = Q_k^{-1} Q_k R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^T A_k Q_k,$$

- Алгоритм сходится последовательно от старших собственных чисел к младшим, 2-3 итерации на одно собственное число.

## QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :
  - Вычисляем QR-разложение:  $A_k = Q_k R_k$
  - Формируем следующую итерацию:  $A_{k+1} = R_k Q_k$

Для симметричных матриц этот процесс сходится к диагональной матрице, содержащей собственные числа. Для произвольных матриц он сходится к верхнетреугольной матрице, где диагональные элементы являются собственными числами исходной матрицы.

$$A_{k+1} = R_k Q_k = Q_k^{-1} Q_k R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^T A_k Q_k,$$

- Алгоритм сходится последовательно от старших собственных чисел к младшим, 2-3 итерации на одно собственное число.
- 1 итерация = подсчёт QR-разложения  $\mathcal{O}(n^3)$  + умножение матриц  $\mathcal{O}(n^3)$ . Наивная сложность QR-алгоритма составляет  $\mathcal{O}(n^4)$ .

## QR-алгоритм

QR-алгоритм выглядит следующим образом:

1. Начинаем с матрицы  $A_0 = A$
2. Для  $k = 0, 1, 2, \dots$  :
  - Вычисляем QR-разложение:  $A_k = Q_k R_k$
  - Формируем следующую итерацию:  $A_{k+1} = R_k Q_k$

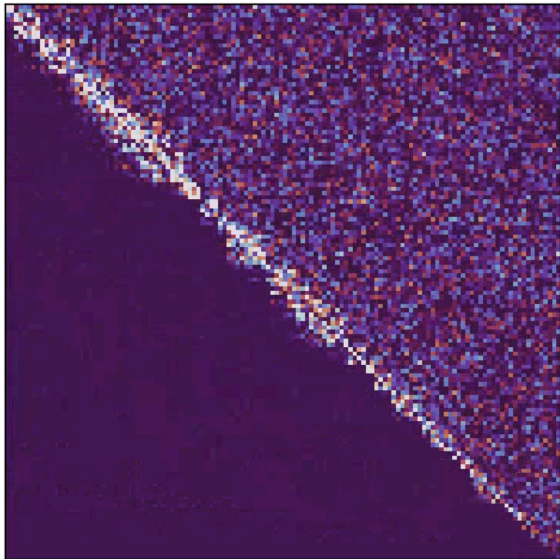
Для симметричных матриц этот процесс сходится к диагональной матрице, содержащей собственные числа. Для произвольных матриц он сходится к верхнетреугольной матрице, где диагональные элементы являются собственными числами исходной матрицы.

$$A_{k+1} = R_k Q_k = Q_k^{-1} Q_k R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^T A_k Q_k,$$

- Алгоритм сходится последовательно от старших собственных чисел к младшим, 2-3 итерации на одно собственное число.
- 1 итерация = подсчёт QR-разложения  $\mathcal{O}(n^3)$  + умножение матриц  $\mathcal{O}(n^3)$ . Наивная сложность QR-алгоритма составляет  $\mathcal{O}(n^4)$ .
- На практике используются различные стратегии ускорения сходимости до  $\mathcal{O}(n^3)$ . Например, приведение матрицы к форме Гессенберга ( $\mathcal{O}(n^3)$ ), QR разложение которой строится за  $\mathcal{O}(n^2)$  итераций. Кроме того, используются сдвиги, которые позволяют ускорить сходимость.

# QR-алгоритм

Случайная матрица



Симметричная матрица @fminxyz

