

# Отчет о выполненном задании по численным методам линейной алгебры.

Егор Подлесов

14 ноября 2023 г.

## Краткий обзор.

Статья показывает результаты реализации и замеров времени работы алгоритма решения **системы линейных алгебраических уравнений** методом **вращений Гивенса**.

Листинг 1: Схема проекта

```
1 | cmc-numerical-methods-of-linear-algebra
2 | -- benchmark
3 |   --- CMakeLists.txt
4 |   --- givens_rotations.cpp
5 | -- CMakeLists.txt
6 | -- src
7 |   | -- CMakeLists.txt
8 |   | -- common.h
9 |   | -- givens_rotations.cpp
10 |   | -- givens_rotations.h
11 |   | -- matrix.cpp
12 |   | -- matrix.h
13 |   | -- triangular_matrix.cpp
14 |   | -- triangular_matrix.h
15 |   | -- vector.cpp
16 |   | -- vector.h
17 | -- tests
18 |   | -- CMakeLists.txt
19 |   | -- givens_rotations_ut.cpp
20 |   | -- SLAU_var_5.csv
```

- **src** - реализация классов **TMatrix**, **TVector**, а также решения **СЛАУ** вращениями Гивенса.
- **benchmark** - замер времени выполнения алгоритма.
- **tests** - тестирование алгоритма.

## Постановка задачи.

**Дано:** Невырожденная матрица  $A$  размеров  $100 \times 100$ .

**Требуется:**

1. Реализовать алгоритм решения **СЛАУ** с коэффициентами - элементами данной матрицы, используя метод вращений Гивенса. Вектор-результат  $b$  сгенерировать из случайных чисел на отрезке  $[-1, 1]$  с равномерным распределением. То есть надо решить систему вида

$$Ax = b,$$

где  $A \in \mathbb{R}^{100 \times 100}$ ,  $b \in \mathbb{R}^{100}$ .

2. Протестировать программу - убедиться что найденный вектор  $x$  удовлетворяет следующему неравенству  $\|Ax - b\| < \varepsilon$ , для некоторого достаточно малого  $\varepsilon$ .
3. Замерить время работы программы на входной матрице.

## Метод вращений Гивенса.

Метод заключается в применении последовательных операций умножения на матрицы специального вида - так называемые матрицы вращений. Вследствии чего исходная матрица получает треугольный вид.

Матрица вращения имеет следующий вид:

$$Q_{ij} = \begin{bmatrix} q_{ii} & \cdots & q_{ij} \\ \vdots & \ddots & \vdots \\ q_{ji} & \cdots & q_{jj} \end{bmatrix}$$

То есть у матрицы  $Q_{ij}$  на всех позициях кроме  $(i, i)$ ,  $(i, j)$ ,  $(j, i)$ ,  $(j, j)$  стоят нулевые элементы. При этом ненулевые элементы подчиняются следующим соотношениям:

$$\begin{cases} q_{ii} = q_{jj} \\ q_{ji} = -q_{ij} \\ q_{ii}^2 + q_{ij}^2 = 1 \end{cases}$$

**Замечание:** Можно положить,  $q_{ii} = \cos(\alpha)$  и  $q_{ij} = \sin(\alpha)$ . Поэтому матрица  $Q_{ij}$  называется матрицей поворота на угол  $\alpha$ .

Таким образом можно занулять поддиагональные элементы в каждой колонке. Для  $j$ -й колонки зануление поддиагонального элемента в строке  $i$  подразумевает матрицу поворота  $Q_{ji}$  с следующими элементами:

$$\begin{cases} q_{jj} = \frac{a_{jj}}{\sqrt{a_{jj}^2 + a_{ij}^2}} \\ q_{ji} = \frac{a_{ij}}{\sqrt{a_{jj}^2 + a_{ij}^2}} \end{cases}$$

Проверив подстановкой, убеждаемся что элемент  $a_{ij}$  зануляется.

Сделав это для всех поддиагональных элементов мы получаем разложение:

$$A = QR$$

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

Система

$$Ax = b$$

получает вид

$$Rx = Q^T b$$

Такая система решается просто. Последовательно находим  $x_n, x_{n-1}, \dots, x_1$ .

## Краткое описание реализации.

Основные функции

- **NTriangularMatrix::SolveSystem** - решение СЛАУ с треугольной матрицей.
- **NGivensRotations::SolveSystem** - решение СЛАУ вращениями Гивенса.
- **NGivensRotations::SystemToTriangular** - приведение матрицы к треугольной форме вращениями Гивенса.

Вся реализация сохранена в репозитории **GitHub: [ypodlesov](#)**

## Тест и замер времени работы программы.

Тест реализован в файле **tests/givens\_rotations\_ut.cpp**. Он проверяет, что вектор решение  $x$  при умножении на матрицу  $A$  отличается от вектора  $b$  по  $l_2$  норме меньше чем на  $\varepsilon = 10^{-6}$ .

На данной в входном файле матрице  $A \in \mathbb{R}^{100 \times 100}$  программа показывает следующие результаты:

Листинг 2: Результаты работы программы

```
1 Run on (32 X 1995.31 MHz CPU s)
2 CPU Caches:
3 L1 Data 32 KiB (x32)
4 L1 Instruction 32 KiB (x32)
5 L2 Unified 4096 KiB (x16)
6 L3 Unified 16384 KiB (x1)
7 Load Average: 0.28, 0.18, 0.11
8 -----
9 Benchmark                Time                CPU      Iterations
10 -----
11 BM_SolveSystem    67812391 ns      67810825 ns          16
```

То есть  $\approx 0.067$  секунды.