«НИУ МЭИ»
ИВТИ
Кафедра математического моделирования

# Лабораторная работа

QR разложение с помощью вращений Гивенса

Работу выполнил:
студент А-16-20 Валеев Рустам

Москва 2023

# 1 Постановка задачи

Цель работы: Решить систему линейных уравнений Ax = b с помощью QR - разложения. Матрица A порядка N имеет лентоную структуру с элементами:

$$A_{i,i} = min\{N, 10\},$$

$$A_{i,i\pm j} = \frac{1}{j},$$

$$i = \overline{0, \mathrm{N}-1},$$

$$j = \overline{1, \mathrm{K}}$$

Элементы вектора b подобраны так, что $x_1 = 1, x_N = 1, x_{1+k} = 1, x_{N-2k} = 1 (k = 1)$, а все остальные элементы равны 0. Для получения матрицы R необходимо использовать вращения Гивенса, применяя классический порядок обхода элементов.

# 2 Теория

Дана СЛАУ Ax = b. Матрицу A можно выразить как:

$$A = QR;$$

Q - ортгональная матрица, т.е. по определению:

$$Q^T = Q^{-1}; (detQ \neq 0);$$

R - верхнетругольная матрица, т.е. с нулевыми значениями под главной диагональю.

Для ортогонального разложения матрицы A воспользуемся вращениями Гивенса.

$$T_{n,n-1}...T_{3,2}T_{n,1}...T_{3,1}T_{2,1}A = R$$

Матрицы $T_{i,j}$ - являются матрицами вращения. Являются модификафией единичной матрицы - едичничная матрица, в которой значения элементов с индексами (i,i), (i,j), (j,i), (j,j) равны соответственно c, s, -s, c, где $c = cos(\theta); s = sin(\theta)$ для некоторого $\theta$
Примеры при N = 5:

$$T_{2,1} = \begin{pmatrix} c & s & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{3,1} = \begin{pmatrix} c & 0 & s & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -s & 0 & c & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Произведение $T_{i,j}$ на произвольную матрицу M слева ($T_{i,j}M$) равносильно повороту вектора равному j-ому стобцу M по часовой стрелки на $\theta$ радиан в координатной плоскости (i, j) (при перемножении изменяются значения i-ой и j-ой строк матрицы.) Т.е. вектор "ложится"на координату i с сохранением длины, значение по j зануляется. Для выполнения этого действия необходимо положить, что

$$c = cos(\theta) = \frac{M_{i,i}}{\sqrt{M_{i,i}^2 + M_{j,i}^2}}$$

$$s = sin(\theta) = \frac{M_{j,i}}{\sqrt{M_{i,i}^2 + M_{j,i}^2}}$$

Таким образом $T_{i,j}M = \tilde{M}$, где $\tilde{M}$ матрица с зануленным элементом с индексом (i, j)

Матрицы $T_{i,j}$ являются ортогональными. Получаем

$$T_{n,n-1}...T_{3,2}T_{n,1}...T_{3,1}T_{2,1}A = R,$$

где $T_{n,n-1}...T_{3,2}T_{n,1}...T_{3,1}T_{2,1} = Q^T$ - так же ортогональна

$$T_{n,n-1}...T_{3,2}T_{n,1}...T_{3,1}T_{2,1} = Q^T$$

$$Q^T A = R| * Q;$$
$$Q^T Q A = RQ;$$

Так как $Q^T$ ортогональна, получаем:

$$Q^T Q A = A;$$

$$A = QR$$

При решении СЛАУ $Ax = b$ имеем:

$$Ax = b \rightarrow \text{вр-я Гивенса } T_{2,1}...T_{n,n-1} \rightarrow Rx = \tilde{b}$$

Далее $x$ находим через обратное ход метода Гаусса.

Так как при каждом действии операторов вращения на матрицу A изменяются только две строки матрицы, то можем явно не перемножать матрицы вращений и следовательно не создавать в явном виде эти матрицы. Вместо этого применить алгоритм к исходной матрице A.

ДЛЯ $i$ от 1 до $N - 1$:
  ДЛЯ $j$ от $i + 1$ до N:
    НАЙТИ $c, s$ по $A_{i,i}$ и $A_{j,i}$
    $i$ -стр = $i$-стр * c + $j$-стр * $s$
    $j$ -стр = $j$-стр * c - $j$-стр * $s$

Где c и s можно найти с помощью алгоритма, устойчивого к малым значениям $a$ и $b$

ЕСЛИ $b = 0$, ТО $c = 1, s = 0$

ИНАЧЕ

ЕСЛИ $|b| > |a|$ ТО

$\tau = \frac{a}{b}; s = \frac{1}{\sqrt{1+\tau^2}}; c = s * \tau;$

ИНАЧЕ

$\tau = \frac{b}{a}; c = \frac{1}{\sqrt{1+\tau^2}}; s = c * \tau;$

# 3 Раздел A

## 3.1 Информация о процессе решения системы при N = 4, При полуширине ленты $K = [N/4]$

```
Матрица A:
[[4. 1. 0. 0.]
 [1. 4. 1. 0.]
 [0. 1. 4. 1.]
 [0. 0. 1. 4.]]

Вектор b:
[5. 6. 5. 1.]

Вектор x:
[1. 1. 1. 0.]

Полуширина ленты
1
sin = 0.242536, cos = 0.970143 для зануления эл-та (0, 1)
Занулен столбец 0
[[4.123106 1.940285 0.242536 0.      ]
 [0.       3.638034 0.970143 0.      ]
 [0.       1.       4.       1.      ]
 [0.       0.       1.       4.      ]]

sin = 0.265043, cos = 0.964237 для зануления эл-та (1, 2)
Занулен столбец 1
[[4.123106 1.940285 0.242536 0.      ]
 [0.       3.772969 1.99562  0.265043]
 [0.       0.       3.599816 0.964237]
 [0.       0.       1.       4.      ]]

sin = 0.267657, cos = 0.963514 для зануления эл-та (2, 3)
Занулен столбец 2
[[4.123106 1.940285 0.242536 0.      ]
 [0.       3.772969 1.99562  0.265043]
 [0.       0.       3.736131 1.999682]
 [0.       0.       0.       3.595973]]

Полученная верхнетреугольная матрица:
[[4.123106 1.940285 0.242536 0.      ]
 [0.       3.772969 1.99562  0.265043]
 [0.       0.       3.736131 1.999682]
 [0.       0.       0.       3.595973]]

Преобразованная правая часть
[6.30592625 5.76858877 3.73613138 0.        ]
```

```
Преобразованная правая часть
[6.30592625 5.76858877 3.73613138 0.        ]
--------------------------------------------------------------------------------
Полученное решение:
[ 1.  1.  1. -0.]

Норма исходного решения 1.7320508075688772
Норма полученного решения 1.7320508075688774
Погрегность во 2-ой норме 2.220446049250313e-16
--------------------------------------------------------------------------------
```

## 3.2 Информация о процессе решения системы при N = 4, при заполненной матрице

```
Матрица A:
[[4.         1.         0.5        0.33333333]
 [1.         4.         1.         0.5       ]
 [0.5        1.         4.         1.        ]
 [0.33333333 0.5        1.         4.        ]]

Вектор b:
[5.5        6.         5.5        1.83333333]

Вектор x:
[1. 1. 1. 0.]

Полуширина ленты
4
--------------------------------------------------------------------------------
sin = 0.242536, cos = 0.970143 для зануления эл-та (0, 1)
sin = 0.120386, cos = 0.992727 для зануления эл-та (0, 2)
sin = 0.08, cos = 0.996795 для зануления эл-та (0, 3)
Занулен столбец 0
[[4.166667 2.08     1.28     0.88    ]
 [0.       3.638034 0.848875 0.404226]
 [0.       0.759144 3.883315 0.939198]
 [0.       0.334673 0.900486 3.942235]]

sin = 0.204269, cos = 0.978915 для зануления эл-та (1, 2)
sin = 0.08969, cos = 0.99597 для зануления эл-та (1, 3)
Занулен столбец 1
[[4.166667 2.08     1.28     0.88    ]
 [0.       3.731434 1.698435 0.938763]
 [0.       0.       3.628036 0.836824]
 [0.       0.       0.751181 3.87365 ]]

sin = 0.202749, cos = 0.979231 для зануления эл-та (2, 3)
Занулен столбец 2
[[4.166667 2.08     1.28     0.88    ]
 [0.       3.731434 1.698435 0.938763]
 [0.       0.       3.704985 1.604821]
 [0.       0.       0.       3.623532]]

Полученная верхнетреугольная матрица:
[[4.166667 2.08     1.28     0.88    ]
 [0.       3.731434 1.698435 0.938763]
 [0.       0.       3.704985 1.604821]
 [0.       0.       0.       3.623532]]

Преобразованная правая часть
[ 7.526667  5.42987   3.704985 -0.      ]
Полученное решение:
[1. 1. 1. 0.]

Норма исходного решения 1.7320508075688772
Норма полученного решения 1.7320508075688772
Погрегность во 2-ой норме 0.0
--------------------------------------------------------------------------------
```

# 4 Раздел B

## 4.1 Информация о процессе решения для больших систем N = 4, 10, 50, 100, 500, 1000, 10000, 100000 При полуширине ленты $K = [N/4]$

```
Порядок матрицы N = 4
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320508075688774
Погрегность во 2-ой норме = 2.220446049250313e-16
Время вычисления в мс = 1
---------------------------------------------------------------------------
Порядок матрицы N = 10
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.732099079213407
Погрегность во 2-ой норме = 4.8271644529807034e-05
Время вычисления в мс = 1
---------------------------------------------------------------------------
Порядок матрицы N = 50
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320560778584948
Погрегность во 2-ой норме = 5.270289617609336e-06
Время вычисления в мс = 26
---------------------------------------------------------------------------
Порядок матрицы N = 100
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320537420108804
Погрегность во 2-ой норме = 2.934442003166282e-06
Время вычисления в мс = 162
Порядок матрицы увеличился в 2.0 раз, время выполнения увеличилось в 6.231411321746806 раз
---------------------------------------------------------------------------
Порядок матрицы N = 500
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.732051294024128
Погрегность во 2-ой норме = 4.864552507477526e-07
Время вычисления в мс = 13628
Порядок матрицы увеличился в 5.0 раз, время выполнения увеличилось в 84.10992761397334 раз
---------------------------------------------------------------------------
Порядок матрицы N = 1000
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.732051010552225
Погрегность во 2-ой норме = 2.029833479078036e-07
Время вычисления в мс = 106737
Порядок матрицы увеличился в 2.0 раз, время выполнения увеличилось в 7.832117676853784 раз
---------------------------------------------------------------------------
```

## 4.2 Информация о процессе решения для больших систем N = 4, 10, 50, 100, 500, 1000, 10000, 100000 при заполненной матрице

```
Порядок матрицы N = 4
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320508075688772
Погрегность во 2-ой норме = 0.0
Время вычисления в мс = 1
--------------------------------------------------------------------------------
Порядок матрицы N = 10
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320508075688774
Погрегность во 2-ой норме = 2.220446049250313e-16
Время вычисления в мс = 2
--------------------------------------------------------------------------------
Порядок матрицы N = 50
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.732050807568878
Погрегность во 2-ой норме = 8.881784197001252e-16
Время вычисления в мс = 98
--------------------------------------------------------------------------------
Порядок матрицы N = 100
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320508075688772
Погрегность во 2-ой норме = 0.0
Время вычисления в мс = 868
Порядок матрицы увеличился в 2.0 раз, время выполнения увеличилось в 8.857065635719144 раз
--------------------------------------------------------------------------------
Порядок матрицы N = 500
Норма исходного решения = 1.7320508075688772
Норма полученного решения = 1.7320508075688803
Погрегность во 2-ой норме = 3.1086244689504383e-15
Время вычисления в мс = 87077
Порядок матрицы увеличился в 5.0 раз, время выполнения увеличилось в 100.31105490567434 раз
```

Из результатов видно, что при увеличении порядка СЛАУ в $n$ раз, время выполнения алгоритма увеличивается примерно в $n^3$ раза, что говорит о том, что сложность алгоритма составляет $O(N^3)$. При этом время работы алгоритма для случая 4.1 (ленточная матрица), значительно меньше, чем для случая 4.2 (полностью заполненной матрицы), поскольку для случая ленты алгоритм не обрабатывает нули за пределами ленты, (кроме одной наддиагонали на лентой, значение, которых меняется после вращений)

Таким образом можно оценить время работы алгоритма для случая ленты при:
$N = 10000 \approx 29,65$ часов
$N = 100000 \approx 3$ года, 5 месяцев

для случая заполненной матрицы при:
$N = 1000 \approx 11,61$ минут

$N = 10000 \approx 8$ дней
$N = 100000 \approx 22$ года

# 5 Листинг

```
import time
import math
import numpy as np


# вспомогательные методы для вывода векторов и матриц
def print_matr(matr):
    to_print = np.empty((matr.shape[0], matr.shape[0]))
    for i in range(matr.shape[0]):
        for j in range(matr[i].shape[0]):
            to_print[i][j] = np.round(matr[i][j], 6)
    print(to_print)
    print()


def print_vector(vect):
    to_print = np.empty(vect.shape[0])
    for i in range(vect.shape[0]):
        to_print[i] = np.round(vect[i], 6)
    print(to_print)
    print()


def create_A_x_b(N, K, k):
    A = np.zeros((N, N))
    for i in range(N):
        A[i][i] = min(N, 10)
        for j in range(1, K + 1):
            if (i - j >= 0):
                A[i][i - j] = 1 / j
            if (i + j < N):
                A[i][i + j] = 1 / j
    x = np.zeros(N)
    x[0] = 1
    x[1] = 1
    x[k - 2] = 1
    x[N - k] = 1
    b = np.dot(A, x)
    return A, x, b

# метод получения синуса и косинуса для поворота вектора
def get_sin_and_cos(a, b):
    c = _
```

```
    s = _
    if (b == 0):
        c = 1
        s = 0
    else:
        if (abs(b) > abs(a)):
            tau = a / b
            s = 1 / math.sqrt(1 + tau ** 2)
            c = s * tau
        else:
            tau = b / a
            c = 1 / math.sqrt(1 + tau ** 2)
            s = c * tau
    return s ,c


# метод не используется
def get_non_zero_rows(matr, k):
    n = matr.shape[0]
    non_zero_matr = [_ for i in range(n)]
    for i in range(n):
        left_ind = i - k
        right_ind = i + k + 2
        if (left_ind < 0 & right_ind >= N):
            non_zero_matr[i] = matr[i]
        elif (left_ind < 0):
            non_zero_matr[i] = matr[i][:right_ind]
        elif (right_ind >= n):
            non_zero_matr[i] = matr[i][left_ind:]
        else:
            non_zero_matr[i] = matr[i][left_ind:right_ind ]
    for n in non_zero_matr:
        print(n)


# врашения Гивенса, на вход принимает исходную матрицу, вектор правой част
# возвращает верхнетреугольную матрицу и измененный вектор b
# метод не обрабатывает нули вне ленты
def givens_rotate(a, b, K, print_data):
    n = a.shape[0]
    a = a.copy()
    b = b.copy()
    for i in range(n - 1):
        for j in range(i + 1, min(n, K + i + 1)):
```

```
            s, c = get_sin_and_cos(a[i][i], a[j][i])
            ai = a[i].copy()
            bi = b[i].copy()
            for l in range(i, min(n, i + K + 2)):
                a[i][l] = a[i][l] * c + a[j][l] * s
                a[j][l] = a[j][l] * c - ai[l] * s
            b[i] = b[i] * c + b[j] * s
            b[j] = b[j] * c - bi * s
            if (print_data):
                print(f'sin = {np.round(s, 6)}, cos = {np.round(c, 6)} для
        if (print_data):
            print(f'Занулен столбец {i}')
            print_matr(a)
    return a, b


# обратный ход метода Гаусса
def gauss_solve(a, b):
    n = b.shape[0]
    x = np.empty(n)
    x[n - 1] = - b[n - 1] / a[n - 1][n - 1]
    diag = n - 2
    for i in range(n - 2, -1, -1):
        s = b[i]
        for j in range(n - 1, diag, -1):
            s -= a[i][j] * x[j]
        x[i] = s / a[i][diag]
        diag -= 1
    return x


# вторая норма вектора
def norm_2(vect):
    norm = 0
    for v in vect:
        norm += v ** 2
    return np.sqrt(norm)


# разница норм двух векторов
def diff_by_norm2(v1, v2):
    return abs(norm_2(v1) - norm_2(v2))
```

```
N = 4
K = N // 4
k = 2

A, x, b = create_A_x_b(N, K, k)
print("Матрица A:")
print(A)
print()
print("Вектор b:")
print(b)
print()
print("Вектор x:")
print(x)
print()
print("Полуширина ленты")
print(K)
print('-----------------------------------------------------------------

R, b_ = givens_rotate(A, b, K, True)
print("Полученная верхнетреугольная матрица:")
print_matr(R)
print("Преобразованная правая часть")
print(b_)
print('-----------------------------------------------------------------

x_ = gauss_solve(R, b_)
print("Полученное решение:")
print_vector(x_)
print(f'Норма исходного решения {norm_2(x)}')
print(f'Норма полученного решения {norm_2(x_)}')
print(f'Погрегность во 2-ой норме {diff_by_norm2(x, x_)}')
print('-----------------------------------------------------------------


N = 4
K = N
k = 2

A, x, b = create_A_x_b(N, K, k)
print("Матрица A:")
print(A)
print()
print("Вектор b:")
```

```
print(b)
print()
print("Вектор x:")
print(x)
print()
print("Полуширина ленты")
print(K)
print('-------------------------------------------------------------------

R, b_ = givens_rotate(A, b, K, True)
print("Полученная верхнетреугольная матрица:")
print_matr(R)
print("Преобразованная правая часть")
print_vector(b_)
print('-------------------------------------------------------------------

x_ = gauss_solve(R, b_)
print("Полученное решение:")
print_vector(x_)
print(f'Норма исходного решения {norm_2(x)}')
print(f'Норма полученного решения {norm_2(x_)}')
print(f'Погрегность во 2-ой норме {diff_by_norm2(x, x_)}')
print('-------------------------------------------------------------------


Ns = [4, 10, 50, 100, 500, 1000, 10 ** 4]
k = 2

prev_runtime = 0
for i in range(len(Ns)) :
    N = Ns[i]
    K = N // 4
    A, x, b = create_A_x_b(N, K, k)
    start = time.time() * 1000.0
    R, b_ = givens_rotate(A, b, K, False)
    runtime = time.time() * 1000.0 - start
    x_  = gauss_solve(R, b_)
    print(f'Порядок матрицы N = {N}')
    print(f'Норма исходного решения = {norm_2(x)}')
    print(f'Норма полученного решения = {norm_2(x_)}')
    print(f'Погрегность во 2-ой норме = {diff_by_norm2(x, x_)}')
    print(f'Время вычисления в мс = {round(runtime)}')
    if (i > 2):
        print(f'Порядок матрицы увеличился в {N / Ns[i - 1]} раз, время вы
```

```
    prev_runtime = runtime
    print('-----------------------------------------------------------------

prev_runtime = 0
for i in range(len(Ns)) :
    N = Ns[i]
    K = N
    A, x, b = create_A_x_b(N, K, k)
    start = time.time() * 1000.0
    R, b_ = givens_rotate(A, b, K, False)
    runtime = time.time() * 1000.0 - start
    x_  = gauss_solve(R, b_)
    print(f'Порядок матрицы N = {N}')
    print(f'Норма исходного решения = {norm_2(x)}')
    print(f'Норма полученного решения = {norm_2(x_)}')
    print(f'Погрегность во 2-ой норме = {diff_by_norm2(x, x_)}')
    print(f'Время вычисления в мс = {round(runtime)}')
    if (i > 2):
        print(f'Порядок матрицы увеличился в {N / Ns[i - 1]} раз, время вы
    prev_runtime = runtime
    print('-----------------------------------------------------------------
```