Содержание

1 Задание

Решить систему линейных уравнений $Ax = b$ с помощью QR-разложения. Матрица A порядка N имеет ленточную структуру с элементами

$$A_{i,i} = \min\{N, 10\},\ A_{i,i\pm j=\frac{1}{j}},\ i = \overline{0, N-1},\ j = \overline{1, K} \tag{1}$$

Метод необходимо реализовать для полуширины ленты $K = [N/2]$, а также для заполненной матрицы ($K = N$). При этом нулевые элементы матрицы (за пределами ленты) обрабатываться не должны.

Элементы вектора правых частей b следует подобрать (вычислить) так, чтобы решением являлся вектор x, в котором $x_1 = 1$, $x_N = 1$, $x_{1+k} = 1$, $x_{N-2k} = 1$, все остальные компоненты равны 0. Здесь k - номер по списку группы.

Следует вычислить абсолютную погрешность решения (в норме $\|\cdot\|_2$) построенных тестовых задач.

Для получения матрицы R использовать вращения Гивенса. Матрицу Q формировать в явном виде не требуется.

Для нечетных вариантов следует применять классический порядок обхода элементов (вращения в плоскостях $(n-1, n), (n-2, n-1), \ldots$,(1,2)). Для четных вариантов следует применять гауссовский порядок обхода элементов (вращения в плоскостях (1,2),(1,3),$\ldots$,(1,n); для следующих столбцов аналогично).

2 Часть А

2.1 Тестовый пример

$K = [N/2], N = 4$

$$A_{4\times4} = \begin{pmatrix} 4 & 1 & 0.5 & 0 \\ 1 & 4 & 1 & 0.5 \\ 0.5 & 1 & 4 & 1 \\ 0 & 0.5 & 1 & 4 \end{pmatrix},\ x = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix},\ b = \begin{pmatrix} 4 \\ 1.5 \\ 1.5 \\ 4 \end{pmatrix} \tag{2}$$

2.2 Синусы и косинусы вращений

№ Шага	sin()	cos()
1	0.	1.
2	0.447214	0.894427
3	0.218218	0.975900
4	0.487950	-0.872872
5	0.261394	0.965232
6	-0.485288	-0.874354

2.3 Промежуточные матрицы

$$T_{2,1}T_{3,2}A = \begin{pmatrix} 3.90360 & 1.60500 & 0.952759 & 0.159125 \\ 0. & 2.88290 & 2.13002 & 0.729205 \\ 0. & -0.894427 & 3.13049 & 0.670820 \\ 0. & 0.5 & 1. & 4. \end{pmatrix} = \tilde{A} \tag{3}$$

$$T_{3,2}T_{4,3}\tilde{A} = \begin{pmatrix} 3.90360 & 1.60500 & 0.952759 & 0.159125 \\ 0. & 2.78267 & 0.925476 & 0.0361480 \\ 0. & 0. & -4.32484 & -2.55439 \\ 0. & 0. & -2.40039 & -3.81881 \end{pmatrix} = \tilde{A}' \tag{4}$$

$$T_{4,3}\tilde{A}' = \begin{pmatrix} 3.90360 & 1.60500 & 0.952759 & 0.159125 \\ 0. & 2.78267 & 0.925476 & 0.0361480 \\ 0. & 0. & 3.78144 & 1.21464 \\ 0. & 0. & 0. & 2.09937 \end{pmatrix} = R \tag{5}$$

2.4 Матрица R и вектор b

$$R = \begin{pmatrix} 3.90360 & 1.60500 & 0.952759 & 0.159125 \\ 0. & 2.78267 & 0.925476 & 0.0361480 \\ 0. & 0. & 3.78144 & 1.21464 \\ 0. & 0. & 0. & 2.09937 \end{pmatrix}, b = \begin{pmatrix} 4.06272 \\ 0.0361480 \\ 1.21464 \\ 2.09937 \end{pmatrix} \tag{6}$$

2.5 Полученное решение

$$x = \begin{pmatrix} 1.0 \\ 2.242574e-17 \\ -1.34979e-16 \\ 1.0 \end{pmatrix} \tag{7}$$

3 Часть Б

3.1 $K = [N/2]$

N	Time	Δ
4	0.0	5.016210086489311e-17
10	0.0009975433349609375	2.8252825077728156e-16
50	0.019946575164794922	9.9412051493466e-16
100	0.08278274536132812	1.1738583099051612e-15
500	2.4753966331481934	2.6125981653517285e-15
1000	11.272654056549072	4.017226741985066e-15

3.2 $K = N$

N	Time	Δ
4	0.0008645057678222656	1.3682964720910556e-16
10	0.001013040542602539	2.971520831290671e-16
50	0.02690911293029785	1.3407876016187622e-15
100	0.10423398017883301	2.0839275120389305e-15
500	3.0123841762542725	4.304584621568834e-15
1000	14.235421895980835	5.899689279940525e-15

Вывод: В ходе выполнения лабораторной работы был реализован алгоритм QR-разложения матрицы. Для получения верхнетреугольной матрицы использовались вращения Гивенса. С помощью QR-разложения были решены СЛАУ с ленточной структурой матрицы A. По вычислительным экспериментам можно оценить временную сложность данного алгоритма разложения:

$$N = 10 \Rightarrow Time \approx 0.000997543,\ N = 100 \Rightarrow Time \approx 0.0827827$$

$$\Downarrow$$

$$Time = \mathcal{O}(T^3)$$

4 Приложение

```python
import numpy as np

num_list_group = 3  # номер в списке группы
N = 4  # размерность системы
K = N // 2  # полуширина ленты
A = np.zeros((N, N))  # матрица A системы
```

```
elmin = min(N, 10)  # элемент главной диагонали
# формируем матрицу A
for i in range(N):
    for j in range(N):
        if i == j:
            A[i][j] = elmin
        if 0 <= i + j <= N - 1 and 1 <= j <= K:
            A[i][i + j] = 1 / j
        if 0 <= i - j <= N - 1 and 1 <= j <= K:
            A[i][i - j] = 1 / j
x = np.zeros(N)  # вектор решения
# формируем вектор решения
for i in range(N):
    if i == 0 or i == N - 1 or (i == num_list_group and 1 <=
     ↪  num_list_group < N - 1) or (
            i == (N - 1) - 2 * num_list_group and 1 <= (N - 1) -
             ↪  2 * num_list_group < N - 1):
        x[i] = 1
B = np.zeros(N)  # вектор правой части
# формируем вектор правой части
B = np.dot(A, x)
A = np.c_[A, B]  # дополняем матрицу системы вектором правой
 ↪  части
# вращения Гивенса
for l in range(N - 1):
    for i in range(N - 1, 0 + l, -1):
        j = i - 1
        if A[i][l] != 0:
            # вычислим коэффициенты C и S
            alem = A[j][l]
            belem = A[i][l]
            C = alem / np.sqrt(alem ** 2 + belem ** 2)
            S = belem / np.sqrt(alem ** 2 + belem ** 2)
            # Произведем вращение =)
            temp =  A[i] * C - A[j] * S
            A[j] = A[j] * C + A[i] * S
            A[i] = temp
B = A[:, N]  # извлекаем вектор правой части из матрицы A
A = np.delete(A, N, 1)  # удаляем из матрицы A добавленный
 ↪  столбец B


# обратный ход метода Гаусса
```

```
46  def Gauss_back_step(A, B):
47      # вектор решения
48      sol = np.zeros(N)
49      for i in range(N - 1, -1, -1):
50          s = 0
51          if i == N - 1:
52              sol[i] = B[i] / A[i][i]
53          else:
54              for j in range(i + 1, N, 1):
55                  s += A[i][j] * sol[j]
56              sol[i] = (B[i] - s) / A[i][i]
57      return sol
```
