



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ «Информатика и системы управления»

КАФЕДРА _____ «Теоретическая информатика и компьютерные технологии»

Летучка № 1
по курсу «Численные методы линейной алгебры»
«Реализация метода прогонки и оценка погрешностей
вычислений»

Студент группы ИУ9-71Б Баев Д.А

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

1. Реализовать метод прогонки для $A \cdot x = f$; A принадлежит $R_{100 \times 100}$; f, x принадлежит R_{100} ; A - трехдиагональная матрица.

2. Найти относительные погрешности метода прогонки в сравнении с идеальным решением, простейшего метода Гаусса в сравнении с идеальным решением и библиотечного метода в сравнении с идеальным решением. Идеальное решение брать как в лабораторной №2.

2 Исходный код

Исходный код программы представлен в листингах 1–??.

Листинг 1 — Реализация метода прогонки

```
1 def thomas(A, b):
2     n = len(b)
3
4     alpha = np.zeros(shape=(n, ))
5     beta = np.zeros(shape=(n, ))
6
7     alpha[0] = -A[0][1] / A[0][0]
8     beta[0] = b[0] / A[0][0]
9
10    for i in range(1, n):
11        if i == n - 1:
12            alpha[i] = 0
13            denominator = A[i][i] + A[i][i - 1] * alpha[i - 1]
14        else:
15            denominator = A[i][i] + A[i][i - 1] * alpha[i - 1]
16            alpha[i] = -A[i][i + 1] / denominator
17        beta[i] = (b[i] - A[i][i - 1] * beta[i - 1]) / denominator
18    x = np.zeros(shape=(n, ))
19    x[n - 1] = beta[n - 1]
20    for i in range(n - 2, -1, -1):
21        x[i] = alpha[i] * x[i + 1] + beta[i]
22    return x
```

Листинг 2 — Реализация вспомогательных функций

```

1 def norm(vector):
2     return np.sqrt(sum(x**2 for x in vector))
3
4 def mul_matrix_by_vector(matrix, vector):
5     assert len(matrix[0]) == len(vector)
6     return np.array([sum(matrix[i][j] * vector[j] for j in range(len(
7         vector))) for i in range(len(matrix))])
8
9 def gauss(A, b):
10     n = len(A)
11     A = deepcopy(A)
12     b = deepcopy(b)
13
14     for i in range(n):
15         if A[i][i] == 0:
16             for j in range(i + 1, n):
17                 if A[j][i] != 0:
18                     A[i], A[j] = A[j], A[i]
19                     break
20
21             for j in range(i + 1, n):
22                 f = A[j][i] / A[i][i]
23                 A[j] -= f * A[i]
24                 b[j] -= f * b[i]
25
26     x = np.zeros(shape=(n, ))
27
28     for i in range(n - 1, -1, -1):
29         x[i] = b[i] / A[i][i]
30         for j in range(i - 1, -1, -1):
31             b[j] -= A[j][i] * x[i]
32
33     return np.array(x)
34
35 def random_tridiagonal_matrix(a, b, n):
36     matrix = np.zeros((n, n))
37     for i in range(n):
38         matrix[i][i] = random.uniform(a, b)
39         if i > 0:
40             matrix[i][i - 1] = random.uniform(a, b)
41         if i < n - 1:
42             matrix[i][i + 1] = random.uniform(a, b)
43
44     return matrix

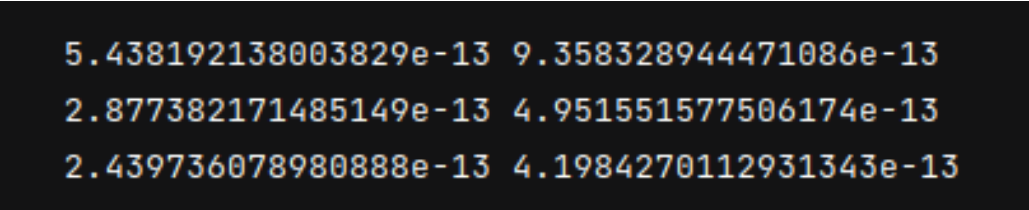
```

Листинг 3 — Оценка погрешностей

```
1 A = random_tridiagonal_matrix(0, 10, 100)
2 x = np.random.uniform(low=0, high=10, size=100)
3 b = mul_matrix_by_vector(A, x)
4
5 x_gauss = gauss(A, b)
6 x_thomas = thomas(A, b)
7 x_np = np.linalg.solve(A, b)
8
9 relative_gauss = norm(x - x_gauss) / norm(x) * 100
10 relative_thomas = norm(x - x_thomas) / norm(x) * 100
11 relative_np = norm(x - x_np) / norm(x) * 100
12
13 print(norm(x - x_gauss), relative_gauss)
14 print(norm(x - x_thomas), relative_thomas)
15 print(norm(x - x_np), relative_np)
```

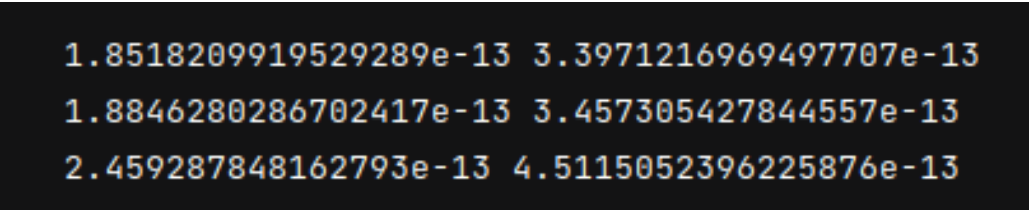
3 Результаты

Результаты оценки погрешностей приведены на рисунках 1- 3.



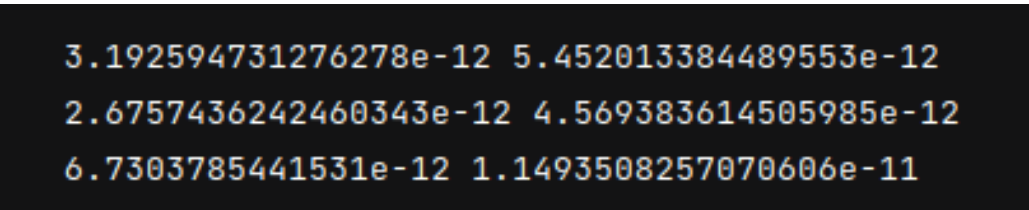
```
5.438192138003829e-13 9.358328944471086e-13
2.877382171485149e-13 4.951551577506174e-13
2.439736078980888e-13 4.1984270112931343e-13
```

Рис. 1 — Результат оценки погрешностей



```
1.8518209919529289e-13 3.3971216969497707e-13
1.8846280286702417e-13 3.457305427844557e-13
2.459287848162793e-13 4.5115052396225876e-13
```

Рис. 2 — Результат оценки погрешностей



```
3.192594731276278e-12 5.452013384489553e-12
2.6757436242460343e-12 4.569383614505985e-12
6.7303785441531e-12 1.1493508257070606e-11
```

Рис. 3 — Результат оценки погрешностей

4 Выводы

В рамках данной лабораторной работы был использован метод прогонки для решения системы линейных уравнений, где матрица была трехдиагональной. Было проведено сравнение относительной погрешности этого метода с аналогичными результатами, полученными при использовании метода Гаусса и функции из библиотеки Numpy. Поскольку метод прогонки специализирован именно для трехдиагональных матриц, его результаты оказались более точными и эффективными по сравнению с методом Гаусса.