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 погрешности вычислений»

Студентка группы ИУ9-72Б Самохвалова П. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель

Реализовать метод прогонки и оценить его погрешность в сравнении с простейшим методом Гаусса и библиотечным решением.

2 Задание

1. Реализовать метод прогонки для Ax = f, A $\in R^{100100}$, f, x $\in R^{100}$, A - трехдиагональная матрица.

2. Найти относительные погрешности метода прогонки, метода Гаусса и библиотечного метода и сравнить их.

3 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Метод прогонки

```
import copy
import random
from num_methods import *
import numpy as np


def progonka_method(m, d):
    n = len(m)
    a = [0]
    b = [0]
    c = [0]
    for i in range(n):
        for j in range(n):
            if i == j:
                b.append(m[i][j])
            elif i == j + 1:
                a.append(m[i][j])
            elif i == j - 1:
                c.append(m[i][j])
    d = [0] + d
    alpha = [0] * n
    beta = [0] * n
    for i in range(1, n):
```

```
        alpha[i] = -c[i] / (a[i - 1] * alpha[i - 1] + b[i])
        beta[i] = (d[i] - a[i - 1] * beta[i - 1]) / (a[i - 1] * alpha[i - 1] + b[i])
    x = [0] * (n + 1)
    x[n] = (d[n] - a[n - 1] * beta[n - 1]) / (a[n - 1] * alpha[n - 1] + b[n])
    for i in range(n - 1, 0, -1):
        x[i] = alpha[i] * x[i + 1] + beta[i]
    x = x[1:]
    return x


def gauss_method(a, b):
    n = len(a)
    for i in range(n):
        for j in range(i + 1, n):
            c = - a[j][i] / a[i][i]
            for k in range(i, n):
                if k == i:
                    a[j][k] = 0
                else:
                    a[j][k] += c * a[i][k]
            b[j] += c * b[i]
    x = [0] * n
    for i in range(n - 1, -1, -1):
        x[i] = b[i]
        for j in range(n - 1, i, -1):
            x[i] -= x[j] * a[i][j]
        x[i] /= a[i][i]
    return x


def generate_tridiagonal_matrix(n, v1, v2):
    a = [[0] * n for i in range(n)]
    for i in range(n):
        if i == 0:
            a[i][i] = random.uniform(v1, v2)
            a[i][i + 1] = random.uniform(v1, v2)
        elif i == n - 1:
            a[i][i] = random.uniform(v1, v2)
            a[i][i - 1] = random.uniform(v1, v2)
        else:
            a[i][i] = random.uniform(v1, v2)
            a[i][i - 1] = random.uniform(v1, v2)
            a[i][i + 1] = random.uniform(v1, v2)
    return a
```

```
68
69
70 n = 100
71 v1 = -100
72 v2 = 100
73 a = generate_tridiagonal_matrix(n, v1, v2)
74 x = [random.uniform(v1, v2) for i in range(1, n + 1)]
75 b = mult_matr_vec(a, x)
76
77 x1 = gauss_method(copy.deepcopy(a), copy.deepcopy(b))
78 print("Gauss method")
79 print(norm_vec(sub_vec(x, x1)) * 100)
80 print()
81
82 x2 = progonka_method(copy.deepcopy(a), copy.deepcopy(b))
83 print("Progonka method")
84 print(norm_vec(sub_vec(x, x2)) * 100)
85 print()
86
87 x3 = np.linalg.solve(a, b)
88 print("Library method")
89 print(norm_vec(sub_vec(x, x3)) * 100)
90 print()
```

4 Результаты

Результаты работы программы представлены на рисунке 1.

5 Выводы

В результате выполнения лабораторной работы был реализован метод прогонки, было проведено сравнение точности решения методом прогонки, методом Гаусса и библиотечным методом.

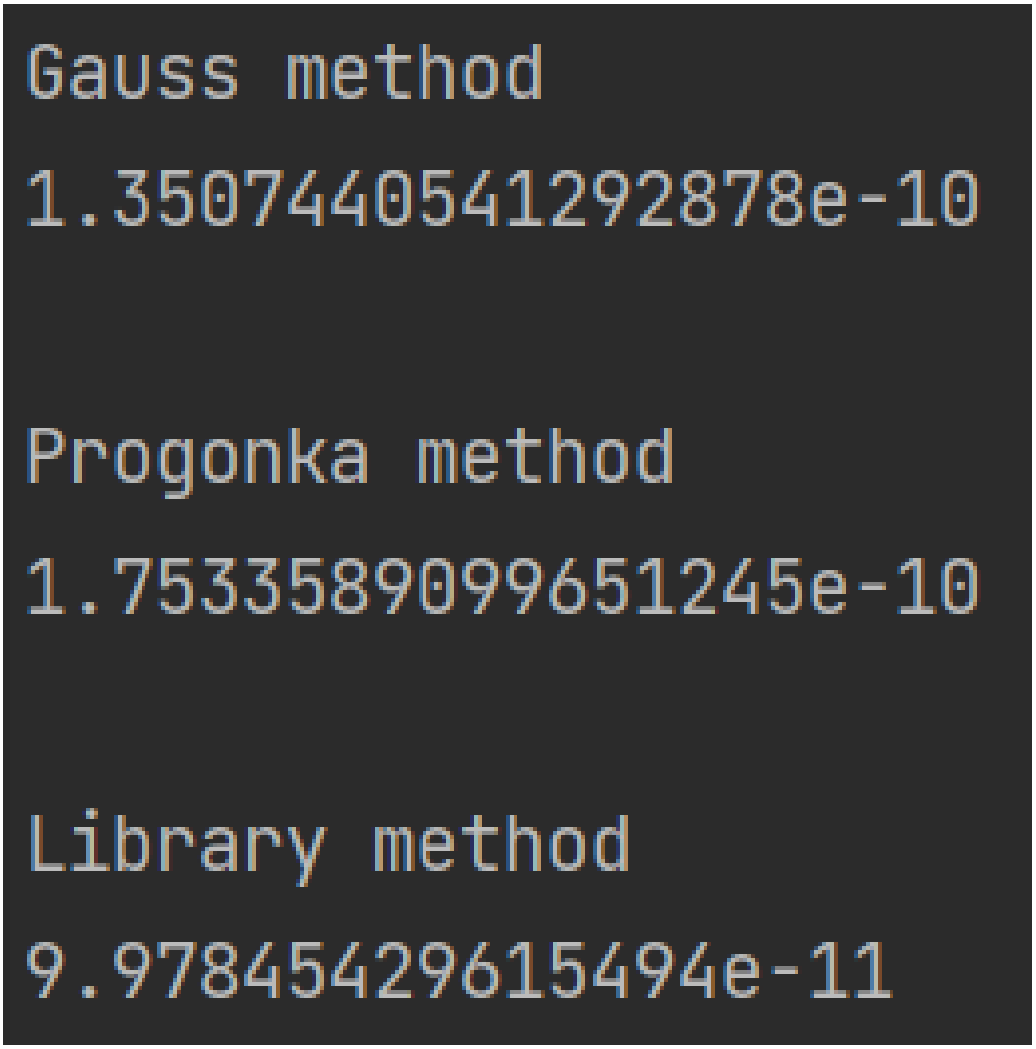

Рис. 1 — Результаты