**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 R100; 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 идеальыом решением. Идеальное решение брать как в лабораторной №2.

# 2 Исходный код

Исходный код программы представлен в листингах 1– **??**.

Листинг 1 — Реализация метода прогонки

```
def thomas(A, b):
    n = len(b)

    alpha = np.zeros(shape=(n, ))
    beta = np.zeros(shape=(n, ))

    alpha[0] = -A[0][1] / A[0][0]
    beta[0] = b[0] / A[0][0]

    for i in range(1, n):
        if i == n - 1:
            alpha[i] = 0
            denominator = A[i][i] + A[i][i - 1] * alpha[i - 1]
        else:
            denominator = A[i][i] + A[i][i - 1] * alpha[i - 1]
            alpha[i] = -A[i][i + 1] / denominator
        beta[i] = (b[i] - A[i][i - 1] * beta[i - 1]) / denominator
    x = np.zeros(shape=(n, ))
    x[n - 1] = beta[n - 1]
    for i in range(n - 2, -1, -1):
        x[i] = alpha[i] * x[i + 1] + beta[i]
    return x
```

Листинг 2 — Реализация вспомогательных функций

```
def norm(vector):
    return np.sqrt(sum(x**2 for x in vector))

def mul_matrix_by_vector(matrix, vector):
    assert len(matrix[0]) == len(vector)
    return np.array([sum(matrix[i][j] * vector[j] for j in range(len(vector))) for i in range(len(matrix))])


def gauss(A, b):
    n = len(A)
    A = deepcopy(A)
    b = deepcopy(b)

    for i in range(n):
        if A[i][i] == 0:
            for j in range(i + 1, n):
                if A[j][i] != 0:
                    A[i], A[j] = A[j], A[i]
                    break

        for j in range(i + 1, n):
            f = A[j][i] / A[i][i]
            A[j] -= f * A[i]
            b[j] -= f * b[i]

    x = np.zeros(shape=(n, ))

    for i in range(n - 1, -1, -1):
        x[i] = b[i] / A[i][i]
        for j in range(i - 1, -1, -1):
            b[j] -= A[j][i] * x[i]

    return np.array(x)

def random_tridiagonal_matrix(a, b, n):
    matrix = np.zeros((n, n))
    for i in range(n):
        matrix[i][i] = random.uniform(a, b)
        if i > 0:
            matrix[i][i-1] = random.uniform(a, b)
        if i < n - 1:
            matrix[i][i+1] = random.uniform(a, b)

    return matrix
```

Листинг 3 — Оценка погрешностей

```
A = random_tridiagonal_matrix(0, 10, 100)
x = np.random.uniform(low=0, high=10, size=100)
b = mul_matrix_by_vector(A, x)

x_gauss = gauss(A, b)
x_thomas = thomas(A, b)
x_np = np.linalg.solve(A, b)

relative_gauss = norm(x - x_gauss) / norm(x) * 100
relative_thomas = norm(x - x_thomas) / norm(x) * 100
relative_np = norm(x - x_np) / norm(x) * 100

print(norm(x - x_gauss), relative_gauss)
print(norm(x - x_thomas), relative_thomas)
print(norm(x - x_np), relative_np)
```

# 3 Результаты

Результаты оценки погрешностей приведены на рисунках 1- 3.

```
5.4381921380038290e-13 9.358328944471086e-13
2.877382171485149e-13 4.951551577506174e-13
2.4397360789808888e-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 4.5693836145059850e-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.