Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1
«Решение СЛАУ»
Вариант №2

Группа P3208

Студент Горин Семён Дмитриевич

Преподаватель Рыбаков Степан Дмитриевич

Санкт-Петербург, 2026

Содержание

1. Цель работы

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них программно.

2. Описание метода

Метод Гаусса.

Основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы. Состоит из 2 фаз: прямого и обратного хода.

Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается x_1 из всех последующих уравнений системы. Затем с помощью второго уравнения исключается x_2 из третьего и всех последующих уравнений и т.д. Этот процесс продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным x_n, т. е. матрица системы будет приведена к треугольному виду.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим неизвестное x_n. Далее, из предыдущего уравнения вычисляем x_{n-1} и т. д. Последним найдем x_1 из первого уравнения.

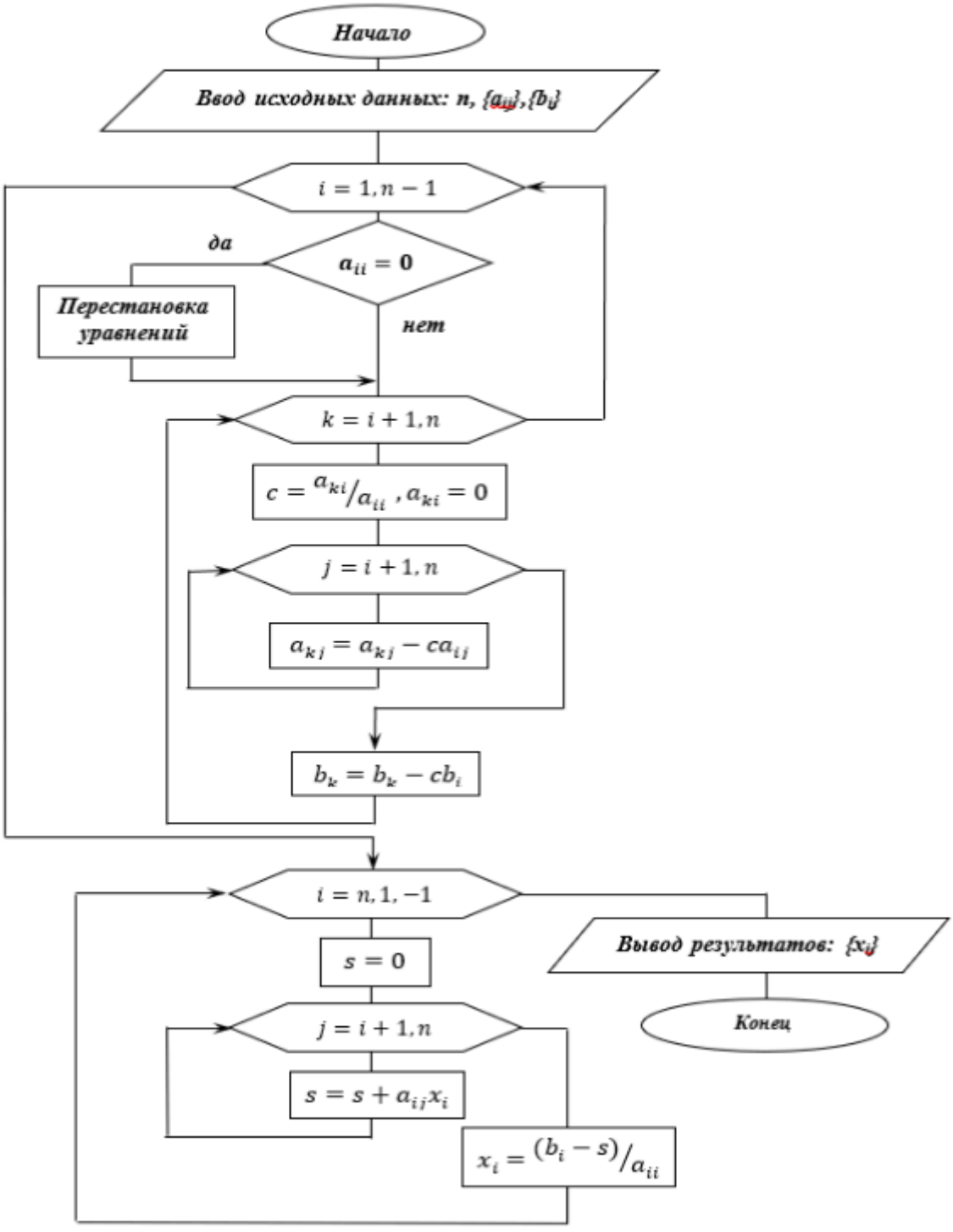

Рис. 1 — Блок-схема метода Гаусса

3. Листинг

Далее приведен листинг функций используемых для нахождения определителя, треугольного представления матрицы коэффициентов, вектора решений и вектора невязок. Также приведены функции с использованием сторонних библиотек для нахождения всего вышеперечисленного.

```
1: def find_determinant(matrix: list[list[float]]) -> float:
2:     """Определитель матрицы"""
3:     n = len(matrix)
4:     matrix = find_triangle_matrix(matrix)
5:     ac = 1
```

```
    for i in range(n):
        ac *= matrix[i][i]
    return ac

def find_triangle_matrix(matrix: list[list[float]]) -> list[list[float]]:
    """Приведение квадратной матрицы или СЛАУ того же ранга к треугольному виду"""
    matrix = [row[:] for row in matrix] # Копируем матрицу
    n = len(matrix)

    for i in range(n):

        pivot_row = None
        for j in range(i, n): # проходимся по строке и ищем по какому из элементов можно приводить матрицу
            if (abs(matrix[j][i]) > 1e-12):
                pivot_row = j
                break

        if (pivot_row != None):
            if (pivot_row != i): # если строка стоит не на своем месте меняем местами
                matrix[pivot_row], matrix[i] = matrix[i], matrix[pivot_row]

            for j in range(i + 1, n): # проходимся по всем строкам ниже и вычитаем из каждой строку по которой приводим
                factor = matrix[j][i] / matrix[i][i] # множитель для конкретной строки
                for k in range(i, len(matrix[j])):
                    matrix[j][k] -= factor * matrix[i][k]
        else:
            raise ValueError("Матрицу невозможно привести к треугольному виду.")

    return matrix


def find_variable_vector(matrix: list[list[float]]) -> list[float]:
    """Нахождение вектора решений"""
    n = len(matrix)

    matrix = [row[:] for row in matrix] # Копируем матрицу
    matrix = find_triangle_matrix(matrix)
    ans = []
    for i in range(n - 1, -1, -1):
        matrix[i][n] /= matrix[i][i]
        for j in range(i - 1, -1, -1):
            matrix[j][n] -= matrix[j][i] * matrix[i][n]
        ans.append(matrix[i][n])
    return ans[::-1]

```

```python
def find_vector_of_residuals(matrix: list[list[float]]) -> list[float]:
    """Нахождение вектора невязок"""
    n = len(matrix)
    matrix = [row[:] for row in matrix] # Копируем матрицу
    variables = find_variable_vector(matrix)
    residuals = [0] * n
    for i in range(n):
        for j in range(n):
                residuals[i] += matrix[i][j] * variables[j]
        residuals[i] -= matrix[i][n]
    return residuals

import numpy as np

def np_find_determinant(matrix: list[list[float]]) -> float:
    A = np.array(matrix, dtype=float)
    A = A[:, : -1]
    return float(np.linalg.det(A))

def np_find_triangle_matrix(matrix: list[list[float]]) ->
list[list[float]]:
    A = np.array(matrix, dtype=float)

    Q, R = np.linalg.qr(A)

    return R.tolist()

def np_find_variable_vector(matrix: list[list[float]]) -> list[float]:
    A = np.array([row[:-1] for row in matrix], dtype=float)
    b = np.array([row[-1] for row in matrix], dtype=float)

    x = np.linalg.solve(A, b)
    return x.tolist()

def np_find_vector_of_residuals(matrix: list[list[float]]) ->
list[float]:
    A = np.array([row[:-1] for row in matrix], dtype=float)
    b = np.array([row[-1] for row in matrix], dtype=float)

    x = np.linalg.solve(A, b)

    residuals = A @ x - b
    return residuals.tolist()
```

4. Примеры работы программы

На следующем листинге представлен пример работы программы с вводом данных из консоли и подсчетом всех параметров при помощи самописных функций:

```
> all_info
```

```
Введите матрицу построчно, каждый новый элемент должен быть отделен от предыдущего пробелом. Для выхода из режима ввода нажмите esc, а затем enter.
Пример ввода:
1 2 3
4 5 6
7 8 9

1 1
Определитель:
1.0
Преобразованнная матрица:
+-----+-----+
| 1.0 | 1.0 |
+-----+-----+
Решение методом Гаусса:
x[1] = 1.0
Вектор невязки:
r[1] = 0.0
```

На следующем листинге представлен пример работы программы с вводом данных из файла и подсчетом всех параметров при помощи библиотеки «numpy»:

```
> all_info --from-file test.txt --numpy
Определитель:
1.0
Преобразованнная матрица:
+-----+-----+
| 1.0 | 1.0 |
+-----+-----+
Решение методом Гаусса:
x[1] = 1.0
Вектор невязки:
r[1] = 0.0
```

5. Вывод

В результате выполнения лабораторной работы я познакомился с численными методами решения СЛАУ, и реализовал метод Гаусса на языке программирования Python.

Сравнивая результат работы программы написанной мной и библиотеки «numpy» можно заметить, что при достаточно несбалансированных матрицах результат может отличаться. Это происходит из-за того что при вычислении необходимых параметров библиотекой «numpy» используется другой численный метод, из-за чего результат операций с числами с плавающей точкой может незначительно отличаться.