

**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

Содержание

1. Цель работы	3
2. Описание метода	3
3. Листинг	4
4. Примеры работы программы	7
5. Вывод	7

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.

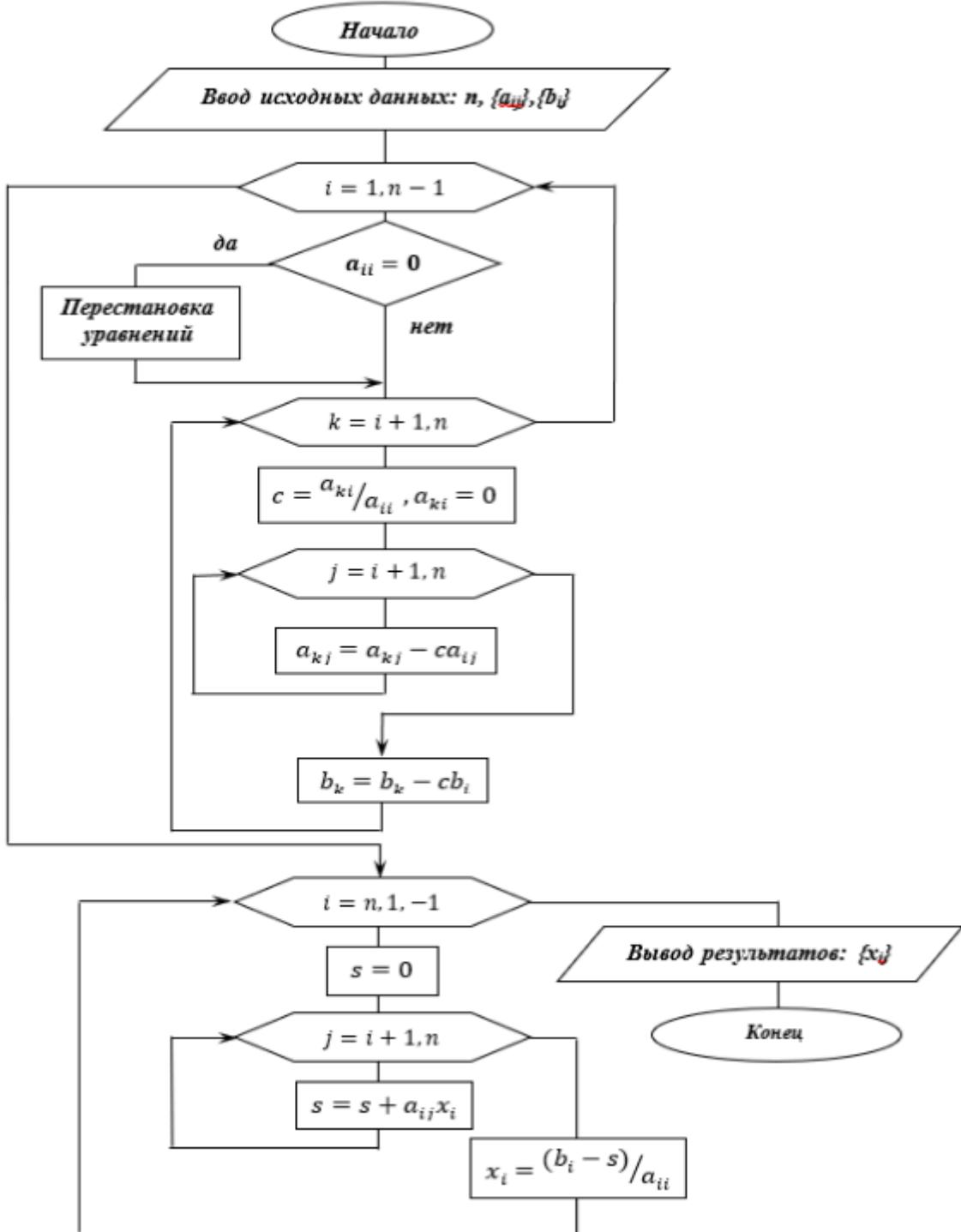


Рис. 1 — Блок-схема метода Гаусса

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

```

6:     triangle_matrix, swap_count = find_triangle_matrix(matrix)
7:
8:
9:     det = 1.0
10:    for i in range(n):
11:        det *= triangle_matrix[i][i]
12:
13:    if swap_count % 2 != 0:
14:        det = -det
15:
16:    return det
17:
18:
19: def find_triangle_matrix(matrix: list[list[float]]) ->
20:     tuple[list[list[float]], int]:
21:         """Приведение квадратной матрицы к треугольному виду.
22:             Возвращает (треугольную матрицу, количество перестановок строк)
23: """
24:         matrix = [row[:] for row in matrix] # Копируем матрицу
25:         n = len(matrix)
26:         swap_count = 0
27:
28:         for i in range(n):
29:
30:             pivot_row = None
31:             for j in range(i, n):
32:                 if abs(matrix[j][i]) > 1e-12:
33:                     pivot_row = j
34:                     break
35:
36:             if pivot_row is None:
37:                 return matrix, swap_count
38:
39:             if pivot_row != i:
40:                 matrix[pivot_row], matrix[i] = matrix[i],
41:                 matrix[pivot_row]
42:                 swap_count += 1
43:
44:                 for j in range(i + 1, n):
45:                     factor = matrix[j][i] / matrix[i][i]
46:                     for k in range(i, n):
47:                         matrix[j][k] -= factor * matrix[i][k]
48:
49:
50: def find_variable_vector(matrix: list[list[float]]) -> list[float]:
51:     """Нахождение вектора решений"""
52:     n = len(matrix)
53:
54:     matrix = [row[:] for row in matrix] # Копируем матрицу
55:     matrix, _ = find_triangle_matrix(matrix)
56:     ans = []

```

```

57:     for i in range(n - 1, -1, -1):
58:         matrix[i][n] /= matrix[i][i]
59:         for j in range(i - 1, -1, -1):
60:             matrix[j][n] -= matrix[j][i] * matrix[i][n]
61:         ans.append(matrix[i][n])
62:     return ans[::-1]
63:
64: def find_vector_of_residuals(matrix: list[list[float]]) ->
65:     list[float]:
66:     """Нахождение вектора невязок"""
67:     n = len(matrix)
68:     matrix = [row[:] for row in matrix] # Копируем матрицу
69:     variables = find_variable_vector(matrix)
70:     residuals = [0] * n
71:     for i in range(n):
72:         for j in range(n):
73:             residuals[i] += matrix[i][j] * variables[j]
74:     residuals[i] -= matrix[i][n]
75:     return residuals
76: import numpy as np
77:
78: def np_find_determinant(matrix: list[list[float]]) -> float:
79:     A = np.array(matrix, dtype=float)
80:     A = A[:, : -1]
81:     return float(np.linalg.det(A))
82:
83: def np_find_triangle_matrix(matrix: list[list[float]]) ->
84:     list[list[float]]:
85:     A = np.array(matrix, dtype=float)
86:     Q, R = np.linalg.qr(A)
87:
88:     return R.tolist()
89:
90: def np_find_variable_vector(matrix: list[list[float]]) -> list[float]:
91:     A = np.array([row[:-1] for row in matrix], dtype=float)
92:     b = np.array([row[-1] for row in matrix], dtype=float)
93:
94:     x = np.linalg.solve(A, b)
95:     return x.tolist()
96:
97: def np_find_vector_of_residuals(matrix: list[list[float]]) ->
98:     list[float]:
99:     A = np.array([row[:-1] for row in matrix], dtype=float)
100:
101:    x = np.linalg.solve(A, b)
102:
103:    residuals = A @ x - b
104:    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:

```
1: > all_info
2: Введите матрицу построчно, каждый новый элемент должен быть отделен от
   предыдущего пробелом. Для выхода из режима ввода нажмите esc, а затем
   enter.
3: Пример ввода:
4: 1 2 3
5: 4 5 6
6: 7 8 9
7:
8: 1 1
9: Определитель:
10: 1.0
11: Преобразованная матрица:
12: +---+---+
13: | 1.0 | 1.0 |
14: +---+---+
15: Решение методом Гаусса:
16: x[1] = 1.0
17: Вектор невязки:
18: r[1] = 0.0
```

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

```
1: > all_info --from-file test.txt --numpy
2: Определитель:
3: 1.0
4: Преобразованная матрица:
5: +---+---+
6: | 1.0 | 1.0 |
7: +---+---+
8: Решение методом Гаусса:
9: x[1] = 1.0
10: Вектор невязки:
11: 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 числен-

ный метод, из-за чего результат операций с числами с плавающей точкой может незначительно отличаться.