

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №1

Вариант 13

Выполнил:

Трикашный М. Д.

P3206

Преподаватель:

Малышева Т. А.

Санкт-Петербург, 2025 г.

Цель работы

Разработать программу для подсчета корней СЛАУ.

Для прямых методов должно быть реализовано:

- Вычисление определителя
- Вывод треугольной матрицы (включая преобразованный столбец В)
- Вывод вектора неизвестных: x_1, x_2, \dots, x_n
- Вывод вектора невязок: r_1, r_2, \dots, r_n

Описание метода

Метод Гаусса с выбором главного элемента — это модификация стандартного метода Гаусса, направленная на повышение числовой стабильности решения системы линейных уравнений.

Прямой ход метода состоит в последовательном исключении неизвестных из уравнений системы с использованием главного элемента, который выбирается для каждой строки в процессе приведения матрицы к верхнетреугольному виду. На каждом шаге выбирается такой элемент в текущем столбце, который имеет наибольшее по модулю значение среди всех оставшихся элементов в этом столбце (главный элемент). Затем строки матрицы меняются местами так, чтобы этот элемент оказался на диагонали. После этого, как и в стандартном методе Гаусса, осуществляется исключение переменных из оставшихся уравнений.

Обратный ход метода аналогичен стандартному методу Гаусса: начиная с последнего уравнения, решаются все уравнения поочередно, вычисляя неизвестные с использованием найденных значений на предыдущих шагах.

Код программы

https://github.com/trikesh32/comp_math

Функция, реализовывающая сам метод

```
def change_lines(matrix, i, j):
    for k in range(len(matrix[0])):
        tmp = matrix[i][k]
        matrix[i][k] = matrix[j][k]
        matrix[j][k] = tmp
    return matrix

def find_line_with_max_element(matrix, i):
    maximum = abs(matrix[i][i])
    res = i
    for j in range(i, len(matrix)):
        if abs(matrix[j][i]) > maximum:
            maximum = matrix[j][i]
            res = j
    print("Максимальный элемент: ", maximum)
    if maximum == 0:
        return -1
    return res

def kill_elements_under(matrix, i):
    for j in range(i + 1, len(matrix)):
        multiplier = matrix[j][i] / matrix[i][i]
        for k in range(i, len(matrix[0])):
            matrix[j][k] -= multiplier * matrix[i][k]
    return matrix

def make_triangle_matrix(matrix):
    k = 0
    for i in range(len(matrix) - 1):
        print(f"Итерация: {i+1}")
        maximum_index = find_line_with_max_element(matrix, i)
        if maximum_index == -1:
            print("Наибольший модуль 0 => на главной диагонали 0 => определитель 0 => нет решений")
            return -1
        print(f"Номер строки с наибольшим модулем: ", maximum_index)
        if maximum_index != i:
            matrix = change_lines(matrix, maximum_index, i)
            print("Меняем местами строки")
            print_matrix(matrix)
            k+=1
        print("Вычтем строки: ")
        matrix = kill_elements_under(matrix, i)
        print_matrix(matrix)
    print("Количество перестановок: ", k)
    return k

def count_determinant(matrix, k):
    res = 1
    for i in range(len(matrix)):
        res *= matrix[i][i]
    return ((-1) ** k) * res

def find_roots(matrix):
    results = []
    for i in range(len(matrix) - 1, -1, -1):
        root = matrix[i][-1]
        for k, j in enumerate(range(i + 1, len(matrix))):
            root -= matrix[i][j] * results[k]
```

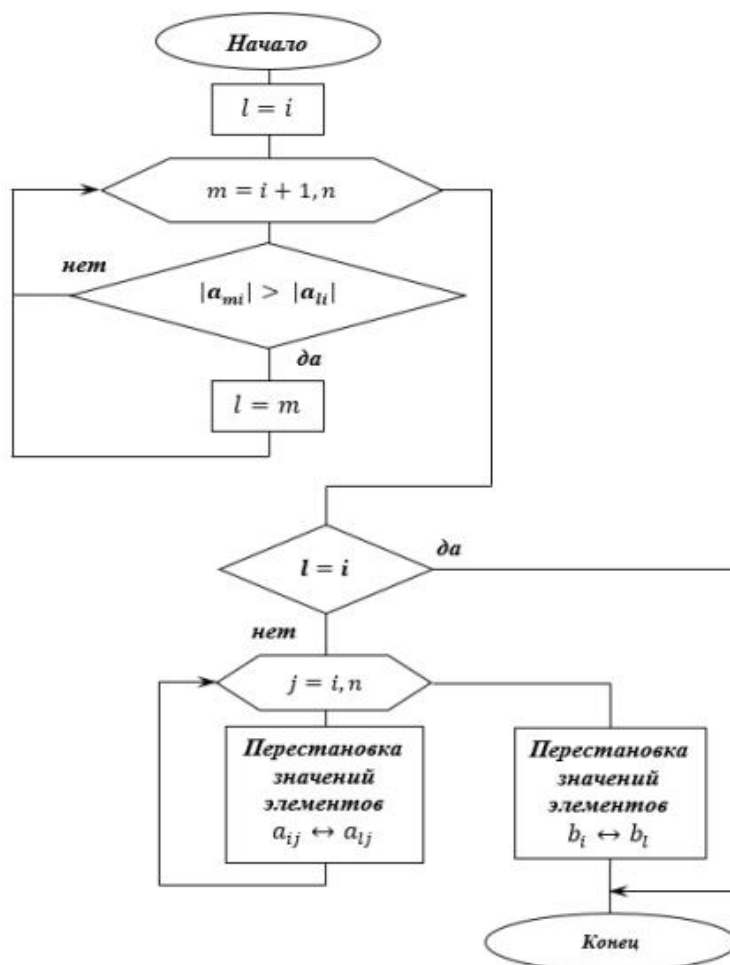
```

    results.insert(0, root / matrix[i][i])
return results

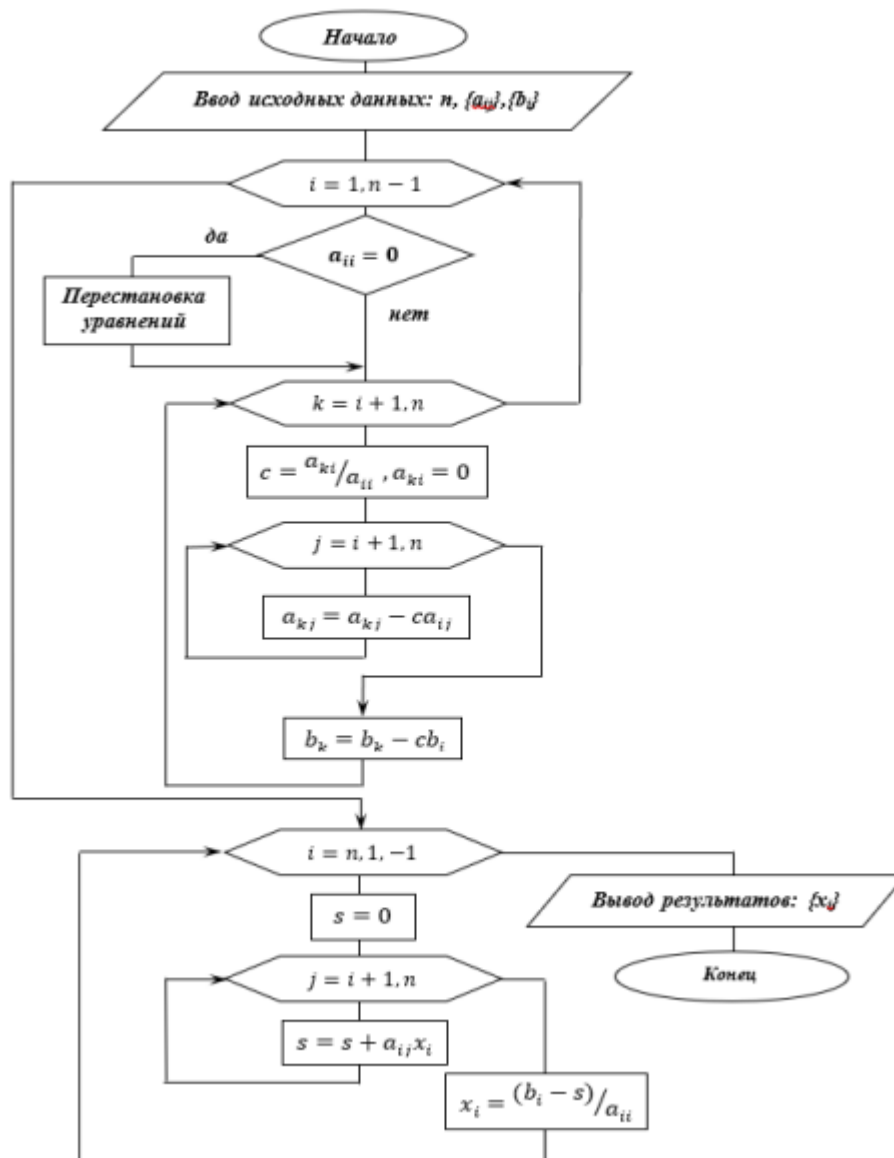
def find_problem_vector(matrix, x):
    result = []
    for i in range(len(matrix)):
        temp = 0
        for j in range(len(matrix)):
            temp += matrix[i][j] * x[j]
        result.append(temp - matrix[i][-1])
    return result

```

Блок схема алгоритма



l – номер наибольшего по абсолютной величине элемента матрицы в столбце с номером i (т. е. среди элементов $a_{ii}, \dots, a_{mi}, \dots, a_{ni}$);
 m – текущий номер элемента, с которым происходит сравнение;
 Выбор главного элемента осуществляется по столбцу



Пример работы программы

Входной файл tests/3.txt:

1	3			
2	2.00	3.00	-1.00	7.00
3	1.00	-1.00	6.00	14.00
4	6.00	-2.00	1.00	11.00

Результат работы:

0. Для выхода с программы

1. Считать матрицу с клавиатуры

2. Считать матрицу с файла

Ваш выбор: 2

Введите путь до файла с матрицей: tests/3.txt

2.000 3.000 -1.000 7.000

1.000 -1.000 6.000 14.000

6.000 -2.000 1.000 11.000

Итерация: 1

Максимальный элемент: 6.0

Номер строки с наибольшим модулем: 2

Меняем местами строки

6.000 -2.000 1.000 11.000

1.000 -1.000 6.000 14.000

2.000 3.000 -1.000 7.000

Вычтем строки:

6.000 -2.000 1.000 11.000

0.000 -0.667 5.833 12.167

0.000 3.667 -1.333 3.333

Итерация: 2

Максимальный элемент: 3.6666666666666665

Номер строки с наибольшим модулем: 2

Меняем местами строки

6.000 -2.000 1.000 11.000

0.000 3.667 -1.333 3.333

0.000 -0.667 5.833 12.167

Вычтем строки:

6.000 -2.000 1.000 11.000

0.000 3.667 -1.333 3.333

0.000 0.000 5.591 12.773

Количество перестановок: 2

Определитель: 123.0

Найденное решение:

2.033 1.740 2.285

Вектор неувязки:

8.881784197001252e-16 -1.7763568394002505e-15 0.0

Найдем решение с помощью библиотеки Numpy

Определитель: 123.00000000000006

[2.03252033 1.7398374 2.28455285]

Вектор неувязки:

[-8.88178420e-16 -1.77635684e-15 0.00000000e+00]

Вывод

Во время выполнения лабораторной работы я изучил работу прямого метода Гаусса с выбором главного элемента по столбцу. Основной недостаток прямого метода – хранение всей матрицы в памяти. Также метод не учитывает количество нулевых элементов, в результате чего проводятся лишние арифметические операции. Из-за того, что результаты вычисления используются повторно, накапливается вычислительная погрешность.

При решении СЛАУ методом Гаусса может получиться большая погрешность из-за использования маленьких ведущих элементов. Выбор главного максимального элемента позволяет избежать этого.

Разница в ответах в моих расчетах и выполненных с помощью библиотеки NumPy может быть вызвана в округлении при выводе на экран. А так векторы неувязки фактически идентичны. Я посмотрел, в NumPy также используются прямые методы, итерационные методы представлены в пакете `scipy.sparse.linalg`.