МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПИИКТ

ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине
‘Вычислительная математика’
Вариант:
Метод Гаусса

Выполнил:
Студент группы P3233
Хасаншин Марат Айратович

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2024

Описание численного метода:

Метод Гаусса является прямым методом для решения систем линейных уравнений.

Пусть нам задано СЛАУ в виде:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

Мы берем первое уравнение и делим его на a_{11} - $x_1 + \frac{a_{12}}{a_{11}}x_2 + \frac{a_{13}}{a_{11}}x_3 = \frac{b_1}{a_{11}}$. После этого, умножив полученный результат на a_{21} и вычтя из второго уравнения мы уберем из него x_1. Аналогично для третьего и a_{31}.

$$\begin{cases} a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 = b_2^{(2)} \\ a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 = b_3^{(2)} \end{cases}$$

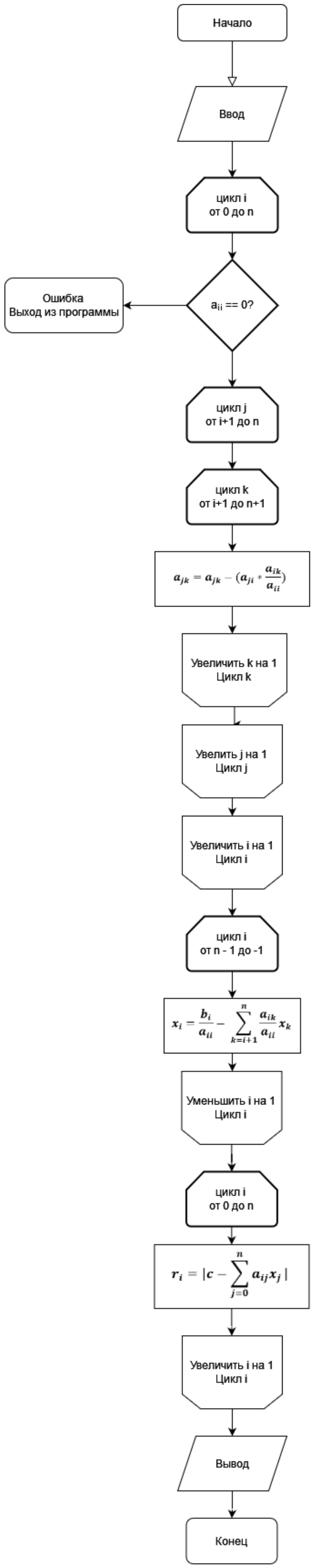
Проделываем эту операцию для всех уравнений. По сути мы строим верхне-диагональную матрицу.

$$a_{33}^{(3)}x_3 = b_3^{(3)}$$

Выражаем отсюда x_3 и подставляем в предыдущую систему. Из нее выражаем уже x_2 и так далее. Это называется обратным ходом.

Так как мы делим на a_{11}, затем на $a_{22}^{(2)}$ и так далее, то важно проверять, не являются ли они 0.

Блок-схема:


Начало
Ввод
цикл i
от 0 до n
a_{ii} == 0?
Ошибка
Выход из программы
цикл j
от i+1 до n
цикл k
от i+1 до n+1
$a_{jk} = a_{jk} - (a_{ji} * \frac{a_{ik}}{a_{ii}})$
Увеличить k на 1
Цикл k
Увелить j на 1
Цикл j
Увеличить i на 1
Цикл i
цикл i
от n - 1 до -1
$x_i = \frac{b_i}{a_{ii}} - \sum_{k=i+1}^{n} \frac{a_{ik}}{a_{ii}} x_k$
Уменьшить i на 1
Цикл i
цикл i
от 0 до n
$r_i = |c - \sum_{j=0}^{n} a_{ij} x_j|$
Увеличить i на 1
Цикл i
Вывод
Конец

Код:

```
def solveByGauss(n, matrix):
    matrix2 = []
    for row in matrix:
        matrix2.append(row[:])
    for i in range(n):
        if (matrix[i][i] == 0):
            Solution.isSolutionExists = False
            return []
        for j in range(i + 1, n):
            for k in range(i + 1, n + 1):
                matrix[j][k] = matrix[j][k] - matrix[j][i] * (matrix[i][k] / 
matrix[i][i])
    answer = []
    for i in range(n-1, -1, -1):
        for j in range(len(answer)):
            matrix[i][n] -= answer[j] * matrix[i][n - j - 1]
        answer.append(matrix[i][n] / matrix[i][i])

    answer.reverse()

    for i in range(n):
        temp = 0
        for j in range(n):
            temp += matrix2[i][j] * answer[j]
        answer.append(abs(temp - matrix2[i][n]))

    return answer
```

Примеры работы программы:

№1:

```
3
3 2 -5 -1
2 -1 3 13
1 2 -1 9
2.9999999999999996
4.999999999999999
3.999999999999999
0.0
3.552713678800501e-15
0.0
```

№2:

```
3
4 2 -1 1
5 3 -2 2
3 2 -3 0
-1.0
3.0
1.0
0.0
0.0
0.0
```

№3:

```
4
2 5 4 1 20
1 3 2 1 11
2 10 9 7 40
3 8 9 2 37
1.0
2.0
2.0
-0.0
0.0
0.0
0.0
0.0
```

№4:

```
3
7 -2 -1 2
6 -4 -5 3
1 2 4 5
The system has no roots of equations or has an infinite set of them.
```

№5:

```
4
2 3 -1 1 1
8 12 -9 8 3
4 6 3 -2 3
2 3 9 -7 3
The system has no roots of equations or has an infinite set of them.
```

Вывод:

В тестах 1, 2 и 3 метод Гаусса выдает корректный ответ с маленькой погрешностью, поскольку корни – целые числа.
В тестах 4 и 5 система не имеет или имеет бесконечное количество решений, и программа корректно это определяет.

По сравнению с другими методами можно выделить следующие плюсы и минусы

Минусы:

- Погрешность быстро растет из-за последовательного расчета и представлений чисел в формате плавающей точки
- Метод неэффективен для больших данных из-за высокой алг. сложности
- Затрачивает много ресурсов на хранение всех данных в память

Плюсы:

- Не требует подготовительных вычислений
- Требует конечного количества операций

Алгоритм может быть применен, если у системы есть решение и нет 0 элементов на главной диагонали.

Алгоритмическая сложность метода – $O(n^3)$, где n – число переменных.

Численная погрешность небольшая при расчете вручную, однако значительно возрастает на компьютере, из-за погрешности при работе с нецелыми числами.