Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2
по дисциплине “Вычислительная математика”
Вариант: Метод Гаусса

Выполнила:
Шевченко Д. П.,
группа P3230

Преподаватель:
Перл О.В.

Санкт-Петербург
2024

Оглавление

Описание численного метода

Для применения метода Гаусса нужно свести матрицу к треугольному виду. Сначала проверяем, не является ли первый элемент (a_{11}) нулем. Потом разделим элементы системы на так называемый ведущий элемент a_{11}, первая строка будет выглядеть так:

$$x_1 + \frac{a_{12}}{a_{11}} x_2 + \cdots + \frac{a_{1n}}{a_{11}} x_n = \frac{b_1}{a_{11}}$$

Исключаем неизвестную x_1 из системы. Для этого из 2,3, ..., n уравнения вычитаем уравнение 1,2,...,n-1 умноженное на $a_{21}, \ldots, a_{n1}$ соответственно. Получаем систему, где коэффициенты рассчитываются

$$a_{ij} = a_{ij} - a_{i1} \frac{a_{1j}}{a_{11}}$$

Продолжаем вычисления пока не останется уравнение с одной неизвестной. Остальные неизвестные выводятся обратной подстановкой, то есть подставляем полученное решение на предыдущем шаге в текущее уравнение.

$$x_{n-1} = \frac{b_{n-1}}{a_{n-1,n-1}} - \frac{a_{n-1,n}}{a_{n-1,n-1}} x_n$$

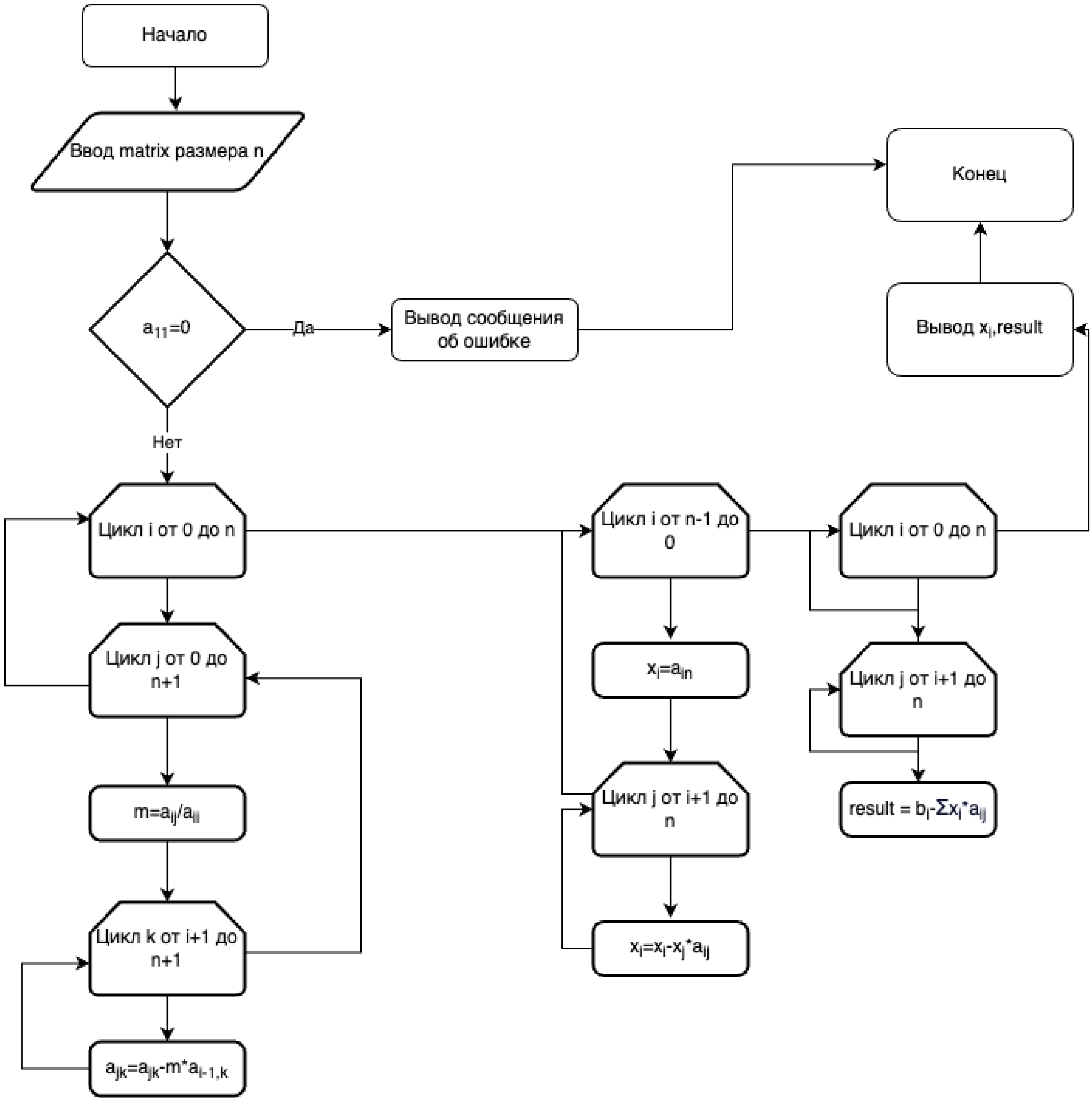
...

$$x_1 = \frac{b_1}{a_{11}} - \frac{a_{1n}}{a_{11}} x_n - \cdots - \frac{a_{12}}{a_{11}} x_2$$

Для расчёта невязок, нужно вычислить разницу между значением справа и значением слева с найденными x.

$$\begin{cases} |a_{11}x_1 + .. + a_{1n}x_n - b_1| = r_1 \\ |a_{nn}x_1 + .. + a_{nn}x_n - b_n| = r_n \end{cases}$$

Блок схема



Начало
Ввод matrix размера n
a11=0
Да
Вывод сообщения об ошибке
Конец
Вывод xi,result
Нет
Цикл i от 0 до n
Цикл j от 0 до n+1
m=aij/aii
Цикл k от i+1 до n+1
ajk=ajk-m*ai-1,k
Цикл i от n-1 до 0
xi=ain
Цикл j от i+1 до n
xi=xi-xj*aij
Цикл i от 0 до n
Цикл j от i+1 до n
result = bi-Σxi*aij

Код

```
class Solution:
    errorMessage = "The system has no roots of equations or has an infinite 
set of them."
    isSolutionExists = True

    def solveByGauss(n, matrix):
        coefficientB = [0] * n
        result = [0] * n
        matrixColumns = n + 1
        initialMatrix = [row[:] for row in matrix]
        for i in range(n):
            coefficientB[i] = matrix[i][n]
        for i in range(n):
            if matrix[i][i] == 0:
                Solution.isSolutionExists = False
                return
            firstRowElement = matrix[i][i]
            for j in range(matrixColumns):
                matrix[i][j] /= firstRowElement
            for j in range(i + 1, n):
                firstInThisRow = matrix[j][i]
                firstElement = matrix[i][i]
                previousRow = i
                for k in range(matrixColumns):
                    matrix[j][k] -= matrix[previousRow][k] * firstInThisRow / 
firstElement
        for i in range(n - 1, -1, -1):
            result[i] = matrix[i][n]
            for j in range(i + 1, n):
                result[i] -= result[j] * matrix[i][j]
        for i in range(n):
            sumLeft = 0
            for j in range(n):
                sumLeft += initialMatrix[i][j] * result[j]
            result.append(coefficientB[i] - sumLeft)

        return result
```

Примеры работы программы

1) Случай, когда коэффициенты пропорциональны друг другу, а свободные члены нет.

```
2
2 3 4
4 6 0
The system has no roots of equations or has an infinite set of them.
```

2) Корректный вывод

```
3
3 2 1 10
1 2 1 8
4 3 -2 4
1.0000000000000009
1.9999999999999991
3.0
0.0
8.881784197001252e-16
0.0
```

3) Случай, когда a_{11}=0

```
2
0 4 6
3 5 8
The system has no roots of equations or has an infinite set of them.
```

4) Матрица 10 на 10

```
10
60 78 31 33 89 43 75 96 22 35 85
89 72 62 50 51 64 76 47 77 87 64
48 24 35 33 16 57 54 99 28 13 22
21 35 57 24 12 98 93 74 18 28 95
13 57 64 57 81 90 85 47 62 66 43
52 71 11 13 81 39 89 80 19 35 60
19 76 54 13 72 61 60 97 75 84 34
26 65 89 73 75 93 55 26 43 81 53
18 39 85 71 99 89 33 83 92 84 11
62 36 49 83 28 20 47 85 35 18 83
0.7950217664918621
-1.4665817876797336
6.542194296905645
-2.8250534430246717
1.4084358435231508
-3.273348004950364
2.453278487544784
-0.5440450671913314
-0.18679784116387488
-2.0041148458666824
2.842170943040401e-14
-2.842170943040401e-14
-1.7763568394002505e-14
2.5579538487363607e-13
3.694822225952521e-13
1.1368683772161603e-13
-3.979039320256561e-13
-2.5579538487363607e-13
-1.0800249583553523e-12
-5.968558980384842e-13
```

5) С нулевым определителем

```
3
1 1 1 6
2 2 2 12
1 1 1 6
The system has no roots of equations or has an infinite set of them.
```

6) Случай, когда был неверный ввод. Нужно добавить обработку этой ситуации, для этого нужно дописать строчки:

```
for i in range(n):
    if len(matrix[i]) != n + 1:
        Solution.isSolutionExists = False
        return []
```

```
2
1 2 2
3 4
Traceback (most recent call last):
  File "/Users/dasha/PycharmProjects/compMath2/mytry.py", line 53, in <module>
    result = Solution.solveByGauss(n, matrix)
  File "/Users/dasha/PycharmProjects/compMath2/mytry.py", line 15, in solveByGauss
    coefficientB[i] = matrix[i][n]
IndexError: list index out of range
```

Правильный вывод:

```
2
1 3 45
2 43
The system has no roots of equations or has an infinite set of them.
```

Выводы

1) Примеры работы программы (страницы 6–8)
2) В методе Гаусса растет вычислительная погрешность, для ее уменьшения используется метод Гаусса с выбором главного элемента. В этом методе угловой элемент выбирается так, чтобы он был максимальным. Но за счет этого растет время выполнения и сложность алгоритма.
   Метод Холецкого имеет меньшую сложность и большую эффективность по сравнению с методом Гаусса, но метод Холецкого подходит для симметричных и положительных матриц, в то время как Гаусса является более универсальным.
3) Анализ применимости метода:
   - эффективен для не очень больших систем, так как для больших будет медленнее работать
   - ведущие элементы должны быть отличны от нуля, то есть главные миноры должны быть ненулевыми
   -применяется в любом случае при наличии решения системы и отсуствия нулей на главной диагонали
4) Анализ алгоритмической сложности:
   Метод Гаусса двухпроходный, имеет прямой и обратный ход.

   Алгоритмическая сложность равна $O(n^3)$, n – количество неизвестных, так как для каждой строки нужно выполнить n-1 линейных преобразований, где у каждого преобразования сложность O(n), итого получаем $O(n*(n-1)*n) = O(n^3)$
5) Анализ численной ошибки:
   Погрешность образуется из-за ограниченной разрядной сетки, в частности с разреженными матрицами. Ошибки копятся из-за последовательного расчета. Численная ошибка вычисляется с помощью невязок. Нужно подставить вычисленное значение неизвестных переменных в уравнение и вычесть то, что получилось из правой части, таким образом мы получим погрешность вычисленных значений.

   Метод Гаусса является хорошим инструментом для матриц ограниченного размера, по сравнению с другими методами он менее трудоемкий.