

Вычислительная математика

Раевский Григорий, группа 3.2

23.03.2024

Отчет по лабораторной работе 2

Содержание

Описание численного метода	2
Диаграмма	3
Листинг кода	5
Примеры работы программы	7
Стандартный случай	7
Нулевая строка	7
Матрица не положительно определенная	8
Несимметричная матрица	8
Матрица с нулями	8
Выводы	10

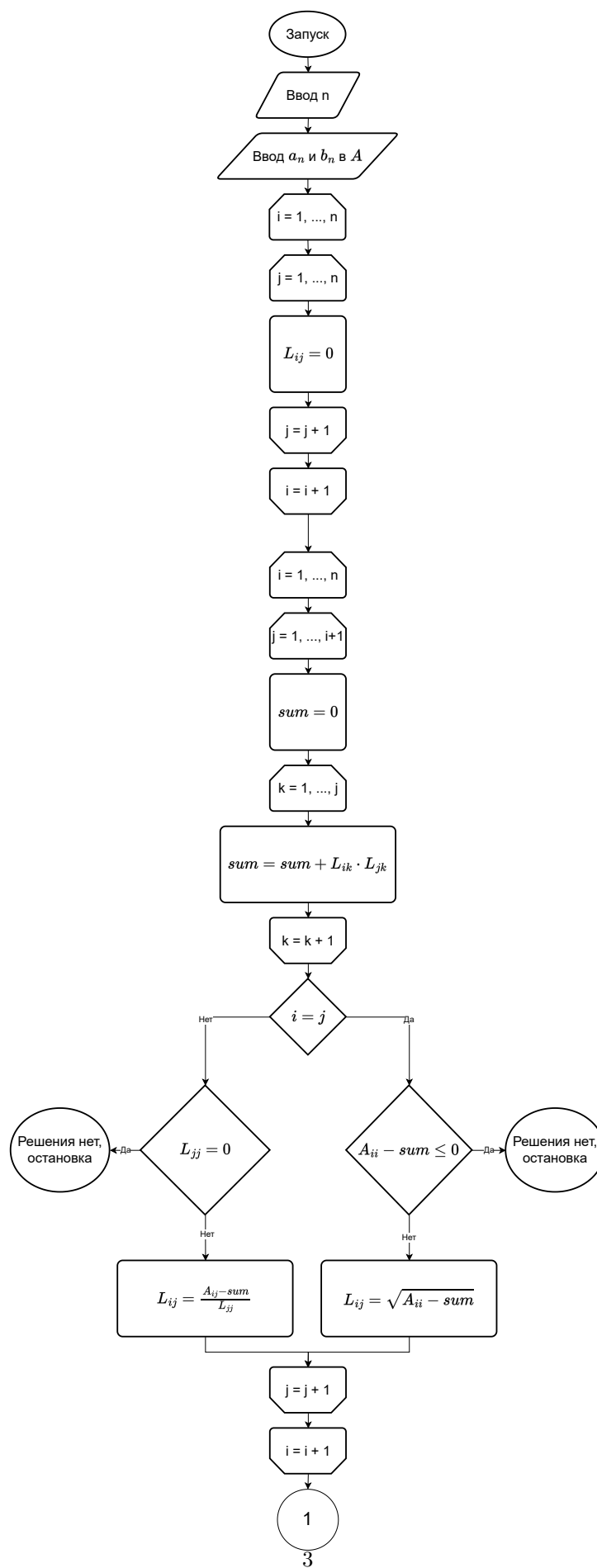
Описание численного метода

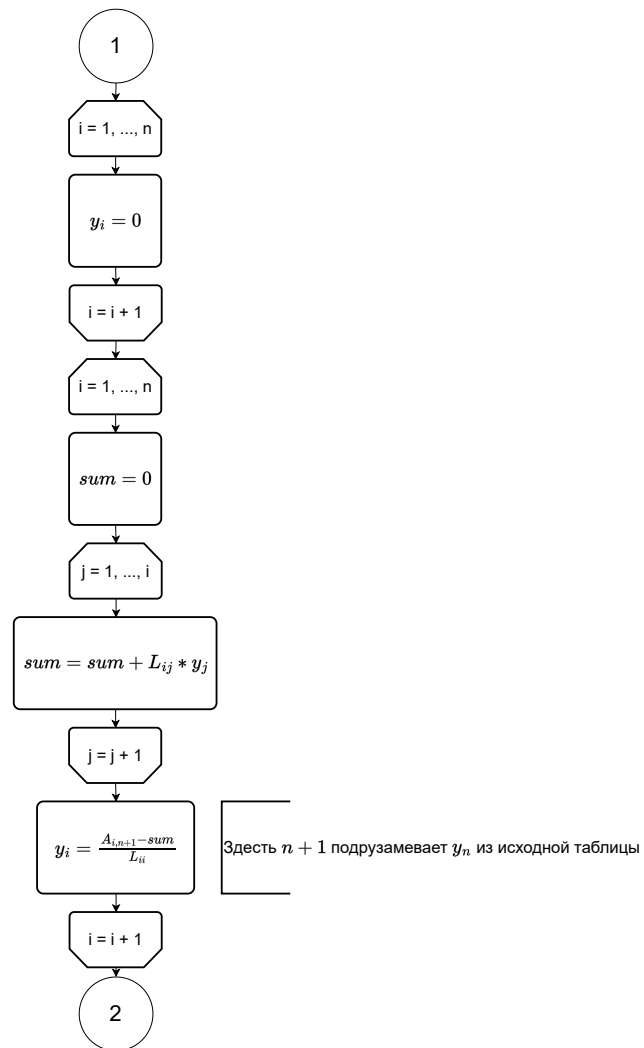
Решение с помощью разложения Холецкого — метод решения системы уравнений с использованием симметричной положительно определенной матрицы. Основан на разложении матрицы A (определяющей коэффициенты в системе СЛАУ) на нижнюю треугольную матрицу L . Таким образом, $A = LL^T$, где L^T — транспонированная треугольная матрица.

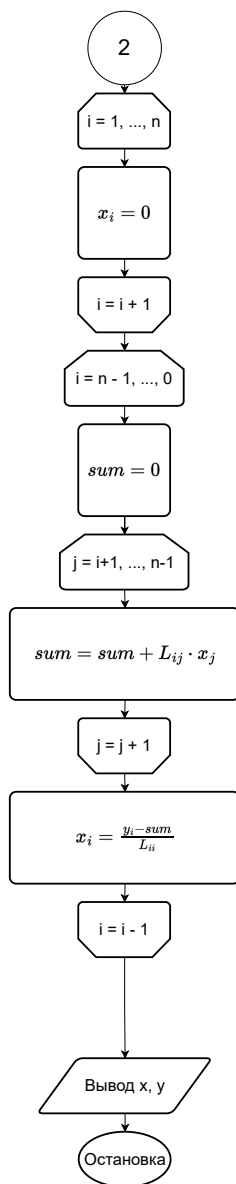
Для каждого элемента l_{ij} суммируется произведения соответствующих пар элементов из предыдущих строк матрицы L , после чего из суммы вычитается a_{ij} . Во время вычисления элементы, стоящие на главной оси матрицы A вычисляются в матрице L , как $l_{ii} = \sqrt{a_{ii} - \text{sum}}$, где l_{ii} - i -й элемент диагонали L матрицы, a_{ii} - диагональный элемент матрицы A , а sum - сумма произведения предыдущих пар элементов матрицы L . Так же, если $i = j$, то необходима проверка матрицы на положительную определенность.

Когда матрица L получена, решаются 2 системы: $Ly = b$ и $L^T x = y$ из которых и получаются исходные значения для x и y в уравнении.

Диаграмма







Листинг кода

```

1  def solveByCholeskyDecomposition(n, matrix):
2      L = [[0.0] * n for _ in range(n)]
3
4      for i in range(n):
5          for j in range(i+1):
6              sum = 0
7              for k in range(j):
8                  sum += L[i][k] * L[j][k]
9              if (i == j):
10                 if (matrix[i][i] - sum <= 0):

```

```

11         Solution.isSolutionExists = False
12         return
13         L[i][j] = math.sqrt(matrix[i][i]-sum)
14     else:
15         if (L[j][j] == 0):
16             Solution.isSolutionExists = False
17             return
18         L[i][j] = (matrix[i][j]-sum) / L[j][j]
19 #print("L: \n")
20 #print(L)
21
22 y = [0 for _ in range(n)]
23 for i in range (n):
24     sum = 0
25     for j in range(i):
26         sum += L[i][j] * y[j]
27     y[i] = (matrix[i][-1]-sum) /L[i][i]
28
29 #print(y)
30
31 x = [0 for _ in range(n)]
32 for i in range(n-1,-1,-1):
33     sum = 0
34     for j in range(i+1,n):
35         sum += L[j][i] *x[j]
36     x[i] = (y[i]-sum)/L[i][i]
37 #print(x)
38 result = x+y
39 return result

```

Примеры работы программы

Стандартный случай

stdin:

```
1 3
2 4 12 -16 1
3 12 37 -43 2
4 -16 -43 98 3
```

stdout:

```
1 28.583333333333332
2 -7.666666666666666
3 1.3333333333333333
4 0.5
5 -1.0
6 4.0
```

Матрица положительная и симметрична \rightarrow метод отработал корректно.

Нулевая строка

stdin:

```
1 2
2 1 0 3
3 0 0 4
```

stdout:

```
1 The system has no roots of equations or has an infinite set of them.
```

2 строка состоит полностью из нулей (за исключением b_2). Это происходит из-за того, что, матрица вырожденная.

Матрица не положительно определенная

stdin:

```
1 2
2 -4 12 2
3 12 37 1
```

stdout:

```
1 The system has no roots of equations or has an infinite set of them.
```

Программа обнаружила, что матрица не является положительно определенной, а значит решение методом Холецкого не возможно.

Несимметричная матрица

stdin:

```
1 2
2 4 16 2
3 12 37 1
```

stdout:

```
1 15.5
2 -5.0
3 1.0
4 -5.0
```

Хоть матрица и не является симметричной, программа отработала корректно и нашла значения x_i и y_i .

Матрица с нулями

stdin:


```
1 4
2 4 0 0 0 1
3 0 1 0 0 2
4 0 0 3 0 3
5 0 0 0 4 4
```

stdout:

```
1 0.25
2 2.0
3 1.0000000000000002
4 1.0
5 0.5
6 2.0
7 1.7320508075688774
8 2.0
```

Программа посчитала значения x_i и y_i даже в матрице с нулями. Это возможно, так как это не противоречит условиям работы алгоритма.

Выводы

Программа стабильно отрабатывает на данных, если они удовлетворяют условиям возможности решения СЛАУ с использованием метода Холецкого. Однако она не способна выполнить расчет, если входные данные нарушают эти требования (например у матрицы с отрицательными элементами невозможно посчитать код для элемента).

По сравнению с методом Холецкого, метод Гаусса менее требователен к матрице, а так же прост в реализации. Однако он обладает высокой алгоритмической сложностью ($O(n^3)$) и может быть неточным из-за накопления ошибок. Метод Холецкого же в зависимости от реализации может быть более быстрым. Он так же имеет меньше проблем с численной стабильностью, так как имеет меньшее количество вычислений. С другой стороны, он более требователен к матрице, так как работает только с симметричными положительными матрицами, а это ограничивает его применимость.

Моя реализация метода обладает алгоритмической сложностью $O(n^3)$ (самая затратная операция — вычисление матрицы L). Метод Холецкого так же