Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №1

по дисциплине «**Вычислительная математика**»

Автор: Баянов Равиль Динарович

Факультет: ПИиКТ

Группа: P3234

Преподаватель: Перл О. В.

Санкт-Петербург, 2024

Оглавление

Описание метода

Метод Холецкого для решения СЛАУ или же разложение Холецкого (метод квадратного корня) – это метод, который позволяет нам разложить матрицу коэффициентов A в виде $A = L^T L$, где L – нижняя треугольная матрица, а L^T – её транспонированная матрица (то есть верхняя треугольная матрица). Важно понимать, что разложение единственно для любой симметричной положительно определённой матрицы.

Алгоритм метода:

Элементы нашей матрицы L можно вычислить по следующим формулам:

$$l_{11} = \sqrt{a_{11}}$$

$$l_{j1} = \frac{a_{j1}}{l_{11}}$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}}{l_{ii}}$$

Затем для того, чтобы использовать это разложение в решении СЛАУ мы должны решить два матричных уравнения последовательно:

$$Ly = b$$

$$L^T x = y$$

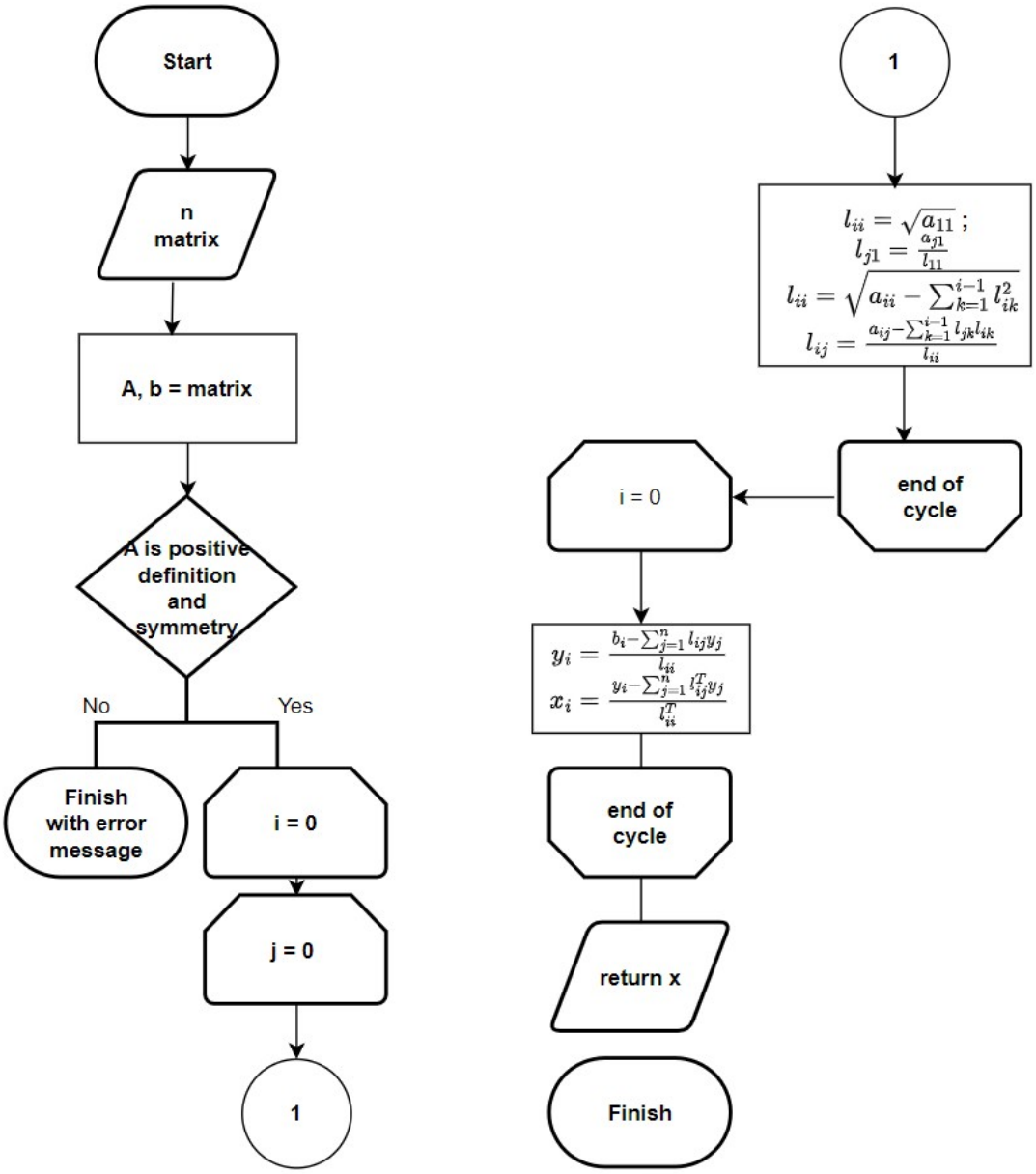
Так как у нас L и L^T треугольные матрицы, то решение системы выше можно свести к таким формулам:

$$(y_i = \frac{\left(b_i - \sum_{i=1}^{n} l_{ij} y_j\right)}{l_{ii}})$$

$$x_i = \frac{\left(y_i - \sum_{i=1}^{n} {l^T}_{ij} y_j\right)}{{l^T}_{ii}})$$

Таким образом, решив эти два уравнения, мы получим вектор x наших корней СЛАУ.

Блок-схема


Start
n
matrix
A, b = matrix
A is positive definition and symmetry
No
Yes
Finish with error message
i = 0
j = 0
1
1
$l_{ii} = \sqrt{a_{11}}$;
$l_{j1} = \frac{a_{j1}}{l_{11}}$
$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$
$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{jk} l_{ik}}{l_{ii}}$
end of cycle
i = 0
$y_i = \frac{b_i - \sum_{j=1}^{n} l_{ij} y_j}{l_{ii}}$
$x_i = \frac{y_i - \sum_{j=1}^{n} l_{ij}^T y_j}{l_{ii}^T}$
end of cycle
return x
Finish

Исходный код метода на языке программирования Python

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

    #
    # Complete the 'solveByCholeskyDecomposition' function
below.
    #
    # The function is expected to return a DOUBLE_ARRAY.
    # The function accepts following parameters:
    #  1. INTEGER n
    #  2. 2D_DOUBLE_ARRAY matrix
    #

    def LUDecomposition(A):
        L = [[0.0] * n for _ in range(n)]
        for i in range(n):
            for j in range(i + 1):
                s = sum(L[i][k] * L[j][k] for k in range(j))
                if i == j:
                    L[i][j] = math.sqrt(A[i][i] - s)
                else:
                    if L[j][j] != 0:
                        L[i][j] = (A[i][j] - s) / L[j][j]
                    else:
                        Solution.isSolutionExists = False
                        return 1
        return L

    def firstEquation(L, b):
        n = len(b)
        y = [0.0] * n
        for i in range(n):
            y[i] = (b[i] - sum(L[i][j] * y[j] for j in
range(i))) / L[i][i]
        return y

    def secondEquation(LT, y):
        n = len(y)
        x = [0.0] * n
        for i in range(n - 1, -1, -1):
            x[i] = (y[i] - sum(LT[i][j] * x[j] for j in range(i
+ 1, n))) / LT[i][i]
        return x

    def matrixSeparation(n, matrix):
        A = []
```

```
        b = []
        for i in range(n):
            A.append([0] * n)
        for i in range(n):
            for j in range(n + 1):
                if j == n:
                    b.append(matrix[i][j])
                else:
                    A[i][j] = matrix[i][j]
        return A, b

    def isPositiveDefinite(matrix):
        n = len(matrix)
        if not all(len(row) == n for row in matrix):
            return False
        if not all(matrix[i][j] == matrix[j][i] for i in
range(n) for j in range(i + 1, n)):
            return False
        return True

    def solveByCholeskyDecomposition(n, matrix):
        A, b = Solution.matrixSeparation(n, matrix)
        if not Solution.isPositiveDefinite(A):
            Solution.isSolutionExists = False
            return 1
        L = Solution.LUDecomposition(A)
        if L == 1:
            return 1
        LT = list(map(list, zip(*L)))
        y = Solution.firstEquation(L, b)
        x = Solution.secondEquation(LT, y)
        result = x + y
        return result
```

Примеры работы программы

1)

```
3
0 2 3 1
0 2 33 4
1 2 3 6
The system has no roots of equations or has an infinite set of them.
```

На этом примере мы видим, что программа не обрабатывает матрицы, который нам не подходят для использования разложение Холецкого. Поэтому выводится сообщение об ошибке. В этом случае у нас матрица не симметрична.

2)

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

Видим, что в этом случае наша матрица она симметрична, но программа всё равно вывела сообщение об ошибке. Значит матрица не является положительно определённой.

3)

```
3
81 -45 45 531
-45 50 -15 -460
45 -15 38 193
6.0
-5.0
-4.0
59.0
-33.0
-12.0
```

Здесь мы видим, что наша матрица симметрична и положительно определена, так как все её собственные числа положительны. Таким образом мы получаем решение нашей СЛАУ. Первые 3 числа - это вектор X, а вторые 3 числа это вектор Y.

4)

```
4
4 2 1 0 5
2 5 2 1 6
1 2 4 2 7
0 1 2 4 0
0.5570469798657719
0.4161073825503357
1.939597315436241
-1.0738255033557043
2.5
1.75
2.485497149149717
-1.8354466280045272
```

Вот пример решения СЛАУ с помощью нашего метода, но уже с матрицей 4 на 4.

5)

```
1
2 2
0.9999999999999999
1.414213562373095
```

Видим, что матрица обрабатывает и решение СЛАУ с матрицей 1 на 1, но выяснилось, что алгоритм теряет немного точность в каких-то вырожденных случаях. И тут мы вместо ожидаемого x = 1 получили 0. 9999999999999999

Вывод

Заметим, что метод Холецкого крайне эффективен. Он выполняется в разы быстрее метода Гаусса. В отличие от метода прогонки, он подходит для большего количества примеров, потому что метод прогонки работает только для решения СЛАУ с трёхдигональной матрицей.

Метод Холецкого выполняется за $O(n^{1/2})$, так как для LU разложения мы проходимся не по всей матрице, а только по её половине. Это делает наш метод быстрым в выполнении.

Трудно оценить ошибку численного метода, так как я в выполнении лабораторной работы не заметил сильных отклонений от правильных ответов. За исключением примера с матрицей 1 на 1. И даже там ошибка не критичная.

Таким образом, метод Холецкого один из лучших методов для решения СЛАУ в силу своей применимости и скорости работы.