Федеральное государственное автономное образовательное
учреждение высшего образования «Национальный
исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа № 2

Разложение Холецкого

Выполнила:

Гафурова Фарангиз Фуркатовна

Группа Р3220

Преподаватель:

Перл Ольга Вячеславовна

г. Санкт-Петербург

2025

Оглавление

Описание метода

Разложение Холецкого – это метод разложения квадратной симметричной положительно определенной матрицы на произведение верхней треугольной матрицы и ее сопряженной (транспонированной, если матрица вещественная).

Основные шаги метода:

Пусть дана квадратная симметричная положительно определенная матрица A. Цель метода — найти верхнюю треугольную матрицу L такая, что,

$$A = LL^T$$

L^T – транспонированная матрица L.

1. Начнем с определения первого элемента l_{11} матрицы L. Из условия $a_{11} = l_{11}^2$ получаем:

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

2. Затем для $j = 2, \ldots, n$ находим элементы первой строки матрицы L по формуле

$$l_{j1} = \frac{a_{j1}}{l_{11}}, \qquad j \in [2, n]$$

3. Для $i = 2, \ldots, n$ выполняем следующие действия:
- Вычисляем:

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \qquad i \in [2, n]$$

- Для $j = i + 1, \ldots, n$ находим

$$l_{ji} = \frac{1}{l_{ii}} \left(a_{ji} - \sum_{k=1}^{i-1} l_{ik} l_{jk} \right), \qquad i \in [2, n-1], j \in [i+1, n].$$

$$Ly = b$$

$$L^T x = y$$

Основные особенности разложения Холецкого

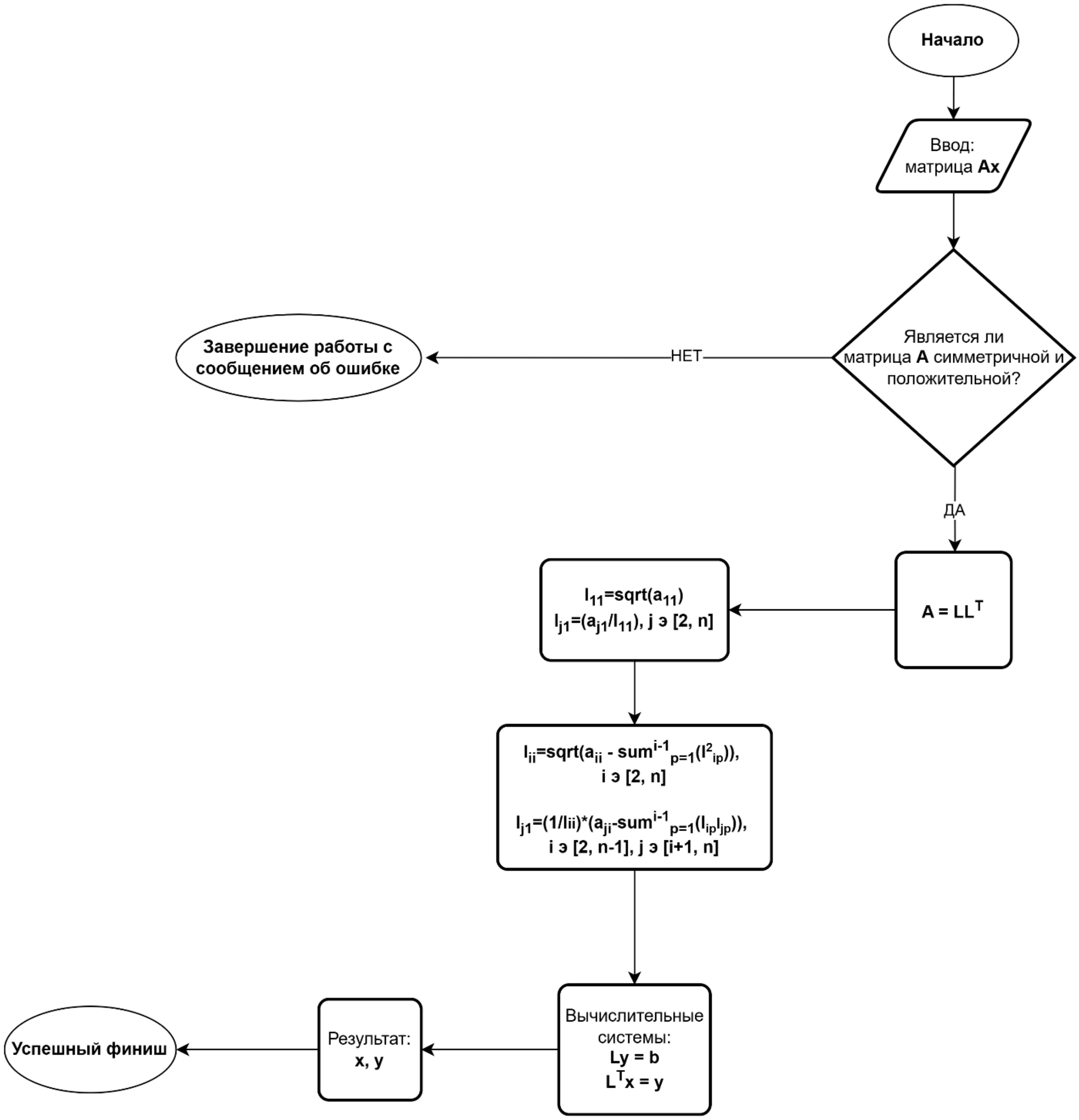
1. **Уникальность**: Для каждой симметричной положительно определенной матрицы A разложение Холецкого существует и является уникальным, при условии, что все диагональные элементы матрицы L положительны.

2. **Треугольная структура**: Результатом разложения является верхняя треугольная матрица L, такая что $A = LL^T$. Это позволяет упростить многие вычисления, поскольку работа с треугольными матрицами проще, чем с произвольными матрицами.

3. **Применение для решения систем линейных уравнений**: Если матрица коэффициентов системы линейных уравнений симметрична и положительно определена, то разложение Холецкого может быть использовано для эффективного решения такой системы. Разложив матрицу коэффициентов на LL^T, можно решить систему в два этапа: сначала найти решение для $Ly = b$ методом прямой прогонки, а затем найти x из уравнения $L^T x = y$ методом обратной прогонки.

4. **Эффективность**: В плане вычислительной сложности разложение Холецкого имеет преимущество перед некоторыми другими методами. Например, для матрицы размерности $n \times n$ требуется порядка $O(n^3)$ операций, что сравнимо с другими эффективными методами разложения матриц, но при этом метод обладает более простой структурой и может быть реализован более легко.

Блок-схема:


Начало
Ввод:
матрица Ax
Является ли
матрица A симметричной и
положительной?
НЕТ
Завершение работы с
сообщением об ошибке
ДА
A = LL^T
l11=sqrt(a11)
lj1=(aj1/l11), j э [2, n]
lii=sqrt(aii - sum^i-1 p=1(l^2 ip)),
i э [2, n]
lj1=(1/lii)*(aji-sum^i-1 p=1(liplјp)),
i э [2, n-1], j э [i+1, n]
Вычислительные
системы:
Ly = b
L^Tx = y
Результат:
x, y
Успешный финиш

Код численного метода (Python)

```python
class Solution:
    isSolutionExists = True
    errorMessage = ""

    #
    # 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 kvaMatrix(n, matrix):
        A = [[0.0] * n for _ in range(n)]
        b = [0.0] * n
        for i in range(n):
            for j in range(n):
                A[i][j] = matrix[i][j]
            b[i] = matrix[i][n]
        return [A, b]

    def solveByCholeskyDecomposition(n, matrix):
        res = Solution.kvaMatrix(n, matrix)
        A = res[0]
        b = res[1]

        Solution.isSimMatrix(n, A)

        return Solution.mainFunc(n, A, b)

    def isSimMatrix(n , A):
        for i in range(n):
            for j in range(n):
                if abs(A[i][j] - A[j][i]) > 1e-12:
                    Solution.isSolutionExists = False
                    Solution.errorMessage = "The system has no roots of equations or has an infinite set of them."
                    return []

    def mainFunc(n,  A, b):
        L = [[0.0] * n for _ in range(n)]
        try:
            for i in range(n):
                for j in range(i + 1):
                    s = 0.0
                    for k in range(j):
                        s += L[i][k] * L[j][k]

                    if i == j:
                        val = A[i][i] - s
                        if val <= 0:
                            Solution.isSolutionExists = False
                            Solution.errorMessage = "The system has no roots of equations or has an infinite set of them."
                            return []
                        L[i][i] = math.sqrt(val)
                    else:
                        L[i][j] = (A[i][j] - s) / L[j][j]
        except:
            Solution.isSolutionExists = False
            Solution.errorMessage = "The system has no roots of equations or has an infinite set of them."
            return []

        y = [0.0] * n
        for i in range(n):
            s = 0.0
            for k in range(i):
                s += L[i][k] * y[k]
            if abs(L[i][i]) < 1e-15:
                Solution.isSolutionExists = False
                Solution.errorMessage = "The system has no roots of equations or has an infinite set of them."
                return []
            y[i] = (b[i] - s) / L[i][i]

        x = [0.0] * n
        for i in range(n - 1, -1, -1):
            s = 0.0
            for k in range(i + 1, n):
                s += L[k][i] * x[k]
            if abs(L[i][i]) < 1e-15:
                Solution.isSolutionExists = False
                Solution.errorMessage = "The system has no roots of equations or has an infinite set of them."
                return []
            x[i] = (y[i] - s) / L[i][i]

        return x + y
```

Тесты:

Тест 1:

Ввод	Вывод
3 24 35 67 23 11 65 34 67 23	(6.58513718461327-2.94345242196767e-16j) (-2.7080748196148754+1.0164924316000427e-16j) (-0.8458109288110687+1.3901038882477378e-16j) 13.676317730539413 (1.4588340760357148e-17-0.23824568472336868j) (-7.664252805791983+6.464909568782445e-16j)

Тест 2:

Ввод	Вывод
4 4 8 13 25 5 2 11 19 7 13 5 25 16 23 29 69	(29.417322834645663+8.466379895772644e-18j) (-0.04724409448818714-3.500786015555017e-17j) (1.6614173228346465-5.128991481542748e-17j) (-6.503937007874015+3.126269905641579e-17j) 12.5 (-3.638500841779361e-16+5.942122813390158j) (-1.962064191641078e-15+17.8978583448784j) (-22.949757503319514-1.633545936398167e-15j)

Тест 3:

Ввод	Вывод
2 109 32 653 266	(0.41565114552797455+1.494965094032666e-17j) (-0.02037668432243374-2.495424123270453e-18j) 3.0650441127076844 (7.533958992334214e-17-1.2303888758096866j)

Тест 4:

Ввод	Вывод
4 452 823 525 25 15 207 192 119 70 143 21 245 162 243 22 659	(0.26347621265171767+4.536950286386627e-17j) (1.900557100991037+1.9486660056207687e-16j) (-2.4860014774936188-1.5455019486736416e-16j) (0.3174104863648852-7.784876951964949e-17j) 1.1759010854794965 8.22330090271549 (1.0626007414329246e-15-17.353587045223957j) (8.06212819914696-1.734275628293575e-15j)

Тест 5:

Ввод	Вывод
5 12 3 7 9 2 4 15 6 2 8 73 7 18 5 1 6 4 2 20 39 9 1 50 4 11	(-1.7159060417337202+1.6094162241690426e-16j) (0.39785035217095227-2.2867185926739556e-17j) (-0.17211268061922377-1.801838472237853e-17j) (1.9114074020340768-3.490604762755356e-17j) (2.455028040568856-3.5005817199659837e-17j) 0.5773502691896258 1.9836731962683511 (-5.397428765838509e-18+0.08814670106673057j) (8.504937189304453-5.879780924689592e-17j) (4.953240933251234-6.302717778927925e-17j)

Тест 6:

Ввод	Вывод
20 34 45 72 34 65 76 23 45 56 23 45 767 68 43 23 76 78 32 54 57 43 74 34 69 29 35 39 45 29 35 32 65 43 23 11 65 67 34 23 74 43 23 54 65 92 38 48 55 65 84 62 14 26 56 48 66 54 48 42 49 46 15 56 68 49 48 59 66 58 54 24 24 15 17 13 18 15 46 15 45 12 3 7 9 2 4 15 6 2 8 73 7 18 5 1 6 4 2 20 39 9 1 50 4 11 452 823 525 25 15 207 192 119 70 143 21 245 162 243 22 659 109 32 653 266 4 8 13 25 5 2 11 19 7 13 5 25 16 23 29 69 4 8 13 25 5 2 11 19 7 13 5 25 16 23 29 69 50 4 11 24 35 67 23 11 65 34 67 23 5 1 6 4 2 20 39 9 1 50 4 11 12 3 7 9 2 4 15 6 2 8 73 7 18 5 1 6 4 2 20 39 24 35 67 23 11 65 34 67 23 5 1 6 4 2 20 39 9 1 50 4 9 1 50 52 82 55 25 15 207 192 11 70 143 21 25 16 43 22 65 32 11 12 3 7 9 2 4 15 6 2 8 73 7 18 5 1 6 4 2 20 39 69 4 8 13 25 5 2 11 19 7 13 5 25 16 23 29 69 50 4 11 11 12 3 7 9 2 4 15 6 2 8 73 7 18 5 1 6 4 2 20 39 15 56 68 49 48 59 66 58 54 24 24 15 17 13 18 15 46 15 45 24 35 67 23 11 65 34 67 23 5 1 6 4 2 20 39 9 1 50 4 69 4 8 13 25 5 2 11 19 7 13 5 25 16 23 29 69 50 4 11 11 12 3 7 9 2 4 15 6 2 8 73 7 18 5 1 6 4 2 20 39 24 35 67 23 11 65 34 67 23 5 1 6 4 2 20 39 9 1 50 4	(-0.2537382496769053-1.1308196044701028e-15j) (3.4719299540867987-2.942648020811803e-16j) (-1.4815451059968383-5.679927913895272e-16j) (-1.0146024213584293+6.905072815179638e-16j) (1.5692948757075484+5.389163886480723e-16j) (0.20124175510142106+2.133659270394722e-18j) (-0.1219261561295459+4.557851263690829e-17j) (-1.1003111325508577+9.122428390180933e-16j) (-0.6968315980998863+5.628453245631938e-19j) (0.7771152892265005-1.0022968162925676e-15j) (-2.0395725637313094+1.023311050712242e-15j) (-0.24867525755309047+4.370398226954367e-17j) (0.44107846286454694+1.2364027025631582e-15j) (0.8441532895839222-7.896571474735486e-16j) (1.429491333800141+1.964124258232764e-15j) (3.129720870716691-6.766561849581393e-16j) (-0.6382573723607051+3.4270150380179567e-16j) (-1.0285140699472524-3.328179329835885e-16j) (-1.5130492577714276+3.0705505388250236e-16j) (0.6729578828940602-4.0359032187013e-16j) 7.37443916112788 (-2.748071138825599e-16+4.487940752776904j) (2.1793053946011933-2.7625253864345788e-17j) (-2.146300826943886e-16+0.29507481889301374j) (23.56661441697498-1.933440559411216e-15j) (-6.176612228917755+8.478225215722845e-16j) (2.0216089091569853e-15+28.737617624281356j) (14.694291365941144-9.349865956454149e-16j) (-1.2140515440510684e-15-9.252312997120482j) (31.561435006671882-3.3704022196485906e-16j) (-1.6907688825545599e-15+4.508201575611806j) (-6.539763718614455e-16-32.68612059273425j) (15.45012251040778-1.2918805917884483e-15j) (3.529578961528665e-16+18.705662068971442j) (42.2568779059072-1.14145096268477e-14j) (1.3482748060295273e-14+35.647373873019326j) (-12.553829051469506-2.3230814564328412e-15j) (-4.5076305312275734e-15+7.761057726118519j) (11.5856955933363-8.388079863655201e-15j) (-8.35565178854678e-15-15.516714141505135j)

Вывод:

В рамках выполнения лабораторной работы мной был изучен и реализован метод разложения Холецкого для решения систем линейных алгебраических уравнений (СЛАУ). Метод Холецкого представляет собой эффективный подход для обработки СЛАУ с симметричными и положительно определенными матрицами коэффициентов.

Алгоритм разложения Холецкого имеет асимптотическую сложность $O(n^3)$, где n - размерность матрицы. Это означает, что с увеличением размерности системы уравнений количество необходимых операций растет кубически. Хотя такая сложность является типичной для многих методов решения СЛАУ, в рамках работы с матрицами, удовлетворяющими условиям симметричности и положительной определенности, метод Холецкого может быть более предпочтителен по сравнению с общими методами, благодаря своей структуре и особенностям вычислений.

Сравнения с другими методами:

- LU - разложение является более универсальным методом, так как оно может быть применено к любой квадратной матрице, независимо от ее симметричности и положительной определенности. Однако, для симметричных положительно определенных матриц метод Холецкого имеет некоторые преимущества.

  Время выполнения: на тестовых матрицах размерности n, проведенные эксперименты показали, что метод Холецкого работает быстрее, чем LU - разложение.

- Метод Гаусса прост в реализации, но имеет асимптотическую сложность $O(n^3)$, аналогичную методу Холецкого. Однако, он менее эффективен для больших матриц, так как не использует особенностей структуры матрицы, как это делает метод Холецкого. Кроме того, метод Гаусса может быть менее точен в некоторых случаях, особенно при наличии близких к нулю элементов на диагонали.
- Методы Якоби и Зейделя являются методами итерационного решения СЛАУ. Они могут быть более подходящими для разреженных матриц, в то время как метод Холецкого лучше подходит для плотных симметричных положительно определенных матриц. Методы Якоби и Зейделя имеют более низкую асимптотическую сложность в некоторых случаях, но могут расходиться при неудачных выборах параметров, в то время как метод Холецкого гарантированно сходится для симметричных положительно определенных матриц.

Метод разложения Холецкого применим только для СЛАУ, у которых матрица коэффициентов A является симметричной ($A_{ij} = A_{ji}$ для всех i и j) и положительно определенной. В случае, если матрица не удовлетворяет этим условиям, алгоритм неприменим. Однако, в некоторых случаях можно преобразовать исходную матрицу. Например, если матрица несимметричная, но имеет положительные собственные значения, можно создать новую симметричную положительно определенную матрицу, которая будет эквивалентна исходной в смысле решения СЛАУ.

Влияние размерности:

При увеличении размерности матрицы численная ошибка может накапливаться. Это связано с тем, что количество операций, в которых могут возникать округления, увеличивается. Например, для матрицы размером 3×3 ошибка округления может быть незначительной и не влиять на результат. Однако, для матрицы 20×20 ошибка может накапливаться на каждом шаге вычислений, что может привести к более заметным отклонениям в результатах. Также при увеличении размерности матрицы увеличивается время выполнения метода, что согласуется с асимптотической сложностью алгоритма.