**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.

Изображение выглядит как черный, темнота

Автоматически созданное описание

Санкт-Петербург, 2024

Оглавление

[Описание метода 3](#_Toc160999556)

[Блок-схема 4](#_Toc160999557)

[Исходный код метода на языке программирования Python 5](#_Toc160999558)

[Примеры работы программы 6](#_Toc160999559)

[Вывод 9](#_Toc160999560)

# 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 = LTL, где L – нижняя треугольная матрица, а LT – 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:

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

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

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

# Блок-схема

Изображение выглядит как диаграмма, зарисовка, текст, рисунок

Автоматически созданное описание

# 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)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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

3)

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

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)

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

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

5)

Изображение выглядит как Шрифт, снимок экрана, текст, Графика

Автоматически созданное описание

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(n1/2), 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.