МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №2

«Метод квадратного корня»

Задание № СЛАУ-04

Выполнил: Снежко Лев Владимирович,

студент 3 курса, 3 группы

Дисциплина: «Численные методы»

Преподаватель: Будник А.М.

Минск, 2024

**Постановка задачи**

Найти решение системы линейных алгебраических уравнений Ax = b с основной матрицей A системы вида:

и столбца свободных членов b вида:

методом квадратного корня. Сравнить точность метода с методом Гаусса.

**Алгоритм решения**

Для преобразования основной матрицы системы к симметрическому виду домножим уравнение Ax = b слева на AT. Получим уравнение вида:

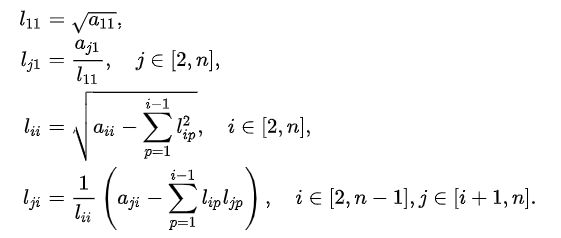
ATAx = ATb

Основная матрица данной системы - ATA. Применим к ней разложение Холецкого, т.е представим в виде:

ATA = LLT

где L – нижняя треугольная матрица.

Для нахождения матрицы L используем формулы:



Теперь уравнение ATAx = ATb принимает вид:

LLTx = ATb

И решение соответствующей СЛАУ сводится к решению следующих СЛАУ:

Ly = ATb

LTx = y

Решим эти СЛАУ с помощью следующих формул (учитывающих, что L – треугольная матрица):

**Листинг**

import numpy as np

import math

N = 5

def cholesky(a):

    #check positive definition

    w, v = np.linalg.eig(np.array(a, dtype='float32'))

    for item in w:

        if item <= 0:

            return

    l=np.array([[0.0] \* N] \* N, dtype='float64')

    l[0, 0] = math.sqrt(a[0, 0])

    for j in range(1, N):

        l[j, 0] = a[j, 0] / l[0, 0]

    for i in range(1, N):

        for j in range(i, N):

            if i == j:

                l[i, i] = math.sqrt(a[i, i] - sum([l[i, p]\*\*2 for p in range(0, i)]))

            else:

                l[j, i] = (a[j, i] - sum([l[i, p] \* l[j, p] for p in range(0, i)])) / l[i,i]

    return l

# solve Ly = b

def find\_y(l, b):

    y = np.zeros(N)

    y[0] = b[0] / l[0,0]

    for i in range(1, N):

        y[i] = (b[i] - np.dot(l[i][:i], y[:i])) / l[i, i]

    return y

#solve L.T \* x = y

def find\_x(l, y):

    l\_t = l.T

    x = np.zeros(N)

    for i in range(N-1, -1, -1):

        x[i] = (y[i] - np.dot(l\_t[i][i+1:], x[i+1:])) / l\_t[i][i]

    return x

#main code

a = np.loadtxt('A.txt', delimiter=' ', dtype='float64')

b = np.loadtxt('B.txt', delimiter=' ', dtype='float64')

a\_sim = a.T @ a

b\_sim = a.T @ b

l = cholesky(a\_sim)

y = find\_y(l, b\_sim)

x = find\_x(l, y)

error = b - a @ x

print(error)

method\_1\_err = np.loadtxt('method\_1\_error.txt', delimiter = ' ', dtype='float64')

diff = error - method\_1\_err

np.savetxt(fname='x.txt', X=x)

np.savetxt(fname='comp\_methods.txt', X=diff)

np.savetxt(fname='method\_2\_error.txt', X=error)

**Результаты и их анализ**

На описанных данных (см. раздел «Постановка задачи») алгоритм выдаёт следующее решение

Х =

Со следующим вектором невязки R:

R = Ax - b

R =

||R|| =

По результатам Лабораторной работы №1 вектор невязки метода Гаусса с выбором опорного элемента по матрице R2 на тех же входных данных:

R2 =

||R2|| =

Таким образом ||R|| > ||R2||. Это связано с тем, что в методе квадратного корня происходит порядка операций против операций метода Гаусса. Таким образом, увеличение невязки можно объяснить большим числом операций, вносящих погрешности. Также такой результат может быть следствием оптимизации метода Гаусса путем выбора опорного элемента по матрице.