, алеМіністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**Звіт**

Про виконання лабораторної роботи №6

**На тему:**

«РОЗВ'ЯЗУВАННЯ ПЕРЕВИЗНАЧЕНИХ СИСТЕМ ЛІНІЙНИХ

АЛГЕБРАЇЧНИХ РІВНЯНЬ»

з дисципліни «чисельні методи»

**Лектор:**

доцент каф. ПЗ

Мельник Н. Б.

**Виконав:**

ст. гр. ПЗ-11

Комарницький В.В.

**Прийняла:**

доцент каф. ПЗ

Мельник Н. Б.

« \_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_\_ \_\_\_\_\_\_\_\_\_ .

Львів – 2022

**Тема.** Розв'язування перевизначених систем лінійних алгебраїчних рівнянь.

**Мета.** Ознайомлення на практиці з методами розв’язування перевизначених систем лінійних алгебраїчних рівнянь.

**Теоретичні відомості**

**Перевизначена система лінійних алгебраїчних рівнянь** – це СЛАР в якій кількість рівнянь є більшою за кількість невідомих. Для того, щоб знайти наближені розв’язки перевизначеної СЛАР використовують метод найменших квадратів(МНК).

Умови мінімізації функції:

Після того, як продиференціюємо функцію за змінними та прирівняємо вирази, які ми визначили на попередньому кроці до нуля, то отримаємо нормальну систему рівнянь:

в якій кількість рівнянь системи дорівнює кількості невідомих.

Щоб отримати нову нормальну систему, скористаємося формулою:

Для того, щоб розв’язати нормальну систему рівнянь скористаємось методом квадратного кореня. Для цього розкладемо матрицю за допомогою методу Холецького, який полягає в тому, що задану матрицю розкладаємо на дві матриці . Елементи першої з них визначаємо за формулами:

Після цього метод поділяється на 2 етапи:

1. Прямий(ми знаходимо значення коренів першої системи рівняння, тобто розв’язки рівняння .
2. Зворотній знаходимо корені нормальної матриці розв’язавши рівняння

**Індивідуальне завдання**

Розв’язати перевизначену систему лінійних алгебраїчних рівнянь

методом найменших квадратів. Отриману відповідну нормальну систему

розв’язати методом квадратного кореня.

Варіант 12.

**Хід роботи**

**Код програми.**

#include <iostream>

#include <iomanip>

#include <vector>

#include <cmath>

**using** std**::**cout**;**

**using** std**::**cin**;**

**using** std**::**endl**;**

**using** std**::**vector**;**

**using** std**::**setw**;**

vector**<**vector**<**double**>>** NormalMatrix**(**vector**<**vector**<**double**>>** vect**,** vector**<**double**>** res**,** int row**,** int column**);**

double determinant**(**vector**<**vector**<**double**>>** vect**,** int size**,** vector**<**double**>** res**);**

int head\_element**(**vector**<**vector **<**double**>>** vect **,** int num\_column**,** int size**);**

void square\_root\_method**(**vector**<**vector**<**double**>>** vect**,** vector**<**double**>** res**,** int row**,** int column**);**

int main**(**void**)**

**{**

int rows**,** columns**;**

cout **<<** "Enter number rows and columns: "**;**

cin **>>** rows **>>** columns**;**

vector**<**vector**<**double**>>** matrix **=** **{{**5**,-**1**,-**3**},**

**{**1**,**2**,-**1**},**

**{**2**,**7**,-**1**},**

**{**1**,-**2**,-**1**},**

**{-**1**,**3**,-**2**}};**

vector**<**double**>** result **=** **{-**1**,**4**,-**1**,**3**,-**2**};**

cout **<<** "\nGiven matrix" **<<** endl**;**

**for(**int i **=** 0**;** i **<** rows**;** **++**i**)**

**{**

**for(**int j **=** 0**;** j **<** columns**;** **++**j**)**

**{**

cout **<<** setw**(**5**)** **<<** matrix**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** setw**(**5**)** **<<** result**[**i**]** **<<** endl**;**

**}**

cout **<<** endl **<<** endl**;**

square\_root\_method**(**matrix**,** result**,** rows**,** columns**);**

**return** 0**;**

**}**

vector**<**vector**<**double**>>** NormalMatrix**(**vector**<**vector**<**double**>>** vect**,** vector**<**double**>** res**,** int row**,** int column**)**

**{**

vector**<**vector**<**double**>>** normMatrix**(**column**,** vector**<**double**>(**column**));**

vector**<**vector**<**double**>>** vect\_transposed**(**column**,** vector **<**double**>(**row**));**

vector**<**double**>** MatrixC**(**column**,** 0.**);**

**for(**int i **=** 0**;** i **<** row**;** **++**i**)**//транспонування заданої матриці коефіцієнтів

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

vect\_transposed**[**j**][**i**]** **=** vect**[**i**][**j**];**

**}**

**}**

cout **<<** "\nMatrix A: " **<<** endl**;**

**for(**int i **=** 0**;** i **<** row**;** **++**i**)**

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

cout **<<** setw**(**5**)** **<<** vect**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

cout **<<** "\nMatrix AT: " **<<** endl**;**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**//вивід транспонованих матриць

**{**

**for(**int j **=** 0**;** j **<** row**;** **++**j**)**

**{**

cout **<<** setw **(**5**)** **<<** vect\_transposed**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

cout **<<** endl**;**

double sum **=** 0.**;**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**//множення транспонованої матриці на задану матрицю

**{**

**for(**int k **=** 0**;** k **<** column**;** **++**k**)**

**{**

sum **=** 0.**;**

**for(**int j **=** 0**;** j **<** row**;** **++**j**)**

**{**

sum **+=** **(**vect\_transposed**[**i**][**j**]** **\*** vect**[**j**][**k**]);**

**}**

normMatrix**[**i**][**k**]** **=** sum**;**

**}**

**}**

cout **<<** "\nNorm Matrix" **<<** endl**;**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**// вивід norm matrix

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

cout **<<** setw**(**5**)** **<<** normMatrix**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

cout **<<** "\nMatrix C" **<<** endl**;**

//search matrix C(C = A^T \* B)

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**

**{**

sum **=** 0.**;**

**for(**int j **=** 0**;** j **<** row**;** **++**j**)**

**{**

sum **+=** **(**vect\_transposed**[**i**][**j**]** **\*** res**[**j**]);**

**}**

MatrixC**[**i**]** **=** sum**;**

**}**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**

**{**

cout **<<** setw**(**5**)** **<<** MatrixC**[**i**]** **<<** endl**;**

**}**

cout **<<** endl **<<** "Determinant = " **<<** determinant**(**normMatrix**,** column**,** res**)** **<<** endl**;**

**if(**determinant**(**normMatrix**,** column**,** res**)** **<** 0**)**

**{**

cout **<<** "Error data!" **<<** endl**;**

**}**

vector**<**vector**<**double**>>** matrix**((**column**),** vector**<**double**>(**column **+** 1**));**

**for(**int i **=** 0**;** i **<** **(**column**);** **++**i**)**

**{**

**for(**int j **=** 0**;** j **<** **(**column**+**1**);** **++**j**)**

**{**

**if(**j **==** column**)**

**{**

matrix**[**i**][**j**]** **=** MatrixC**[**i**];**

**continue;**

**}**

matrix**[**i**][**j**]** **=** normMatrix**[**i**][**j**];**

**}**

**}**

**return** matrix**;**

**}**

double determinant**(**vector**<**vector**<**double**>>** vect**,** int size**,** vector**<**double**>** res**)**//пошук детермінанта за допопмогою зведення до трикутної матриці, тобто прямий хід

**{**

double deter **=** 1.**;**

int num\_permutations **=** 0**;**

double h\_element**;**//головний елемент

**for(**int i **=** 0**;** i **<** **(**vect**.**size**()** **-** 1**);** **++**i**)**

**{**

int index\_head **=** head\_element**(**vect**,** i**,** size**);**

**if(**index\_head **!=** **-**2**)**//якщо не рівне 0, то викликаємо функцію swap і пізніше прирівнюємо головний елемент елементу головної діагоналі

**{**

**for(**int k **=** i**;** k **<** size**;** **++**k**)**

**{**

double tmp **=** vect**[**i**][**k**];**

vect**[**i**][**k**]** **=** vect**[**index\_head**][**k**];**

vect**[**index\_head**][**k**]** **=** tmp**;**

**}**

double tmp1 **=** res**[**i**];**

res**[**i**]** **=** res**[**index\_head**];**

res**[**index\_head**]** **=** tmp1**;**

**++**num\_permutations**;**

h\_element **=** vect**[**i**][**i**];**

**}**

**else** //якщо функція повертає 0, то головним є елемент головної діагоналі

**{**

h\_element **=** vect**[**i**][**i**];**

**}**

int k **=** 0**;**

**for(**k **=** **(**i **+** 1**);** k **<** 3**;** **++**k**)**

**{**

double a **=** vect**[**k**][**i**]** **/** h\_element**;**

**for(**int j **=** i**;** j **<** 3**;** **++**j**)**

**{**

vect**[**k**][**j**]** **=** vect**[**k**][**j**]** **-** vect**[**i**][**j**]** **\*** a**;**

**}**

res**[**k**]** **=** res**[**k**]** **-** res**[**i**]** **\*** a**;**

**}**

**}**

**for(**int i **=** 0**;** i **<** size**;** **++**i**)**// знаходження визначника

**{**

**for(**int j **=** 0**;** j **<** size**;** **++**j**)**

**{**

**if(**i **==** j**)**

**{**

deter **=** deter **\*** vect**[**i**][**j**];**

**}**

**}**

**}**

**if(**num\_permutations **%** 2 **!=** 0**)**

**{**

deter **\*=** **-**1**;**

**}**

**return** deter**;**

**}**

int head\_element**(**vector**<**vector **<**double**>>** vect **,** int num\_column**,** int size**)**//num\_column = iteration index

**{**

int position **=** num\_column**;**

double head\_el **=** abs**(**vect**[**num\_column**][**num\_column**]);**

**for(**int i **=** num\_column**;** i **<** size**;** **++**i**)**

**{**

**if(**head\_el **<** abs**(**vect**[**i**][**num\_column**]))**

**{**

head\_el **=** vect**[**i**][**num\_column**];**

position **=** i**;**

**}**

**}**

**if(**position **!=** num\_column**)**

**{**

**return** position**;**

**}**

**return** **-**2**;**

**}**

void square\_root\_method**(**vector**<**vector**<**double**>>** vect**,** vector**<**double**>** res**,** int row**,** int column**)**

**{**

vector**<**vector**<**double**>>** matrixN **=** NormalMatrix**(**vect**,** res**,** row**,** column**);**

vector**<**vector**<**double**>>** L**(**column**,** vector**<**double**>(**column**,** 0**));**//matrix L

vector**<**vector**<**double**>>** LT**(**column**,** vector**<**double**>(**column**,** 0**));**//matrix L transponovana

cout **<<** "\nNew system of equations" **<<** endl**;**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**

**{**

**for(**int j **=** 0**;** j **<** column**+**1**;** **++**j**)**

**{**

cout **<<** setw**(**10**)** **<<** matrixN**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

double sum **=** 0.**;**

L**[**0**][**0**]** **=** sqrt**(**matrixN**[**0**][**0**]);**//перший елемент

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**//матриця L - нижньотрикутна матриця

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

sum **=** 0.**;**

**if(**j **==** 0**)**//заповнення першого стовпця

**{**

L**[**i**][**j**]** **=** matrixN**[**i**][**j**]** **/** L**[**0**][**0**];**

**continue;**

**}**

**if(**i **==** j**)**//діагональні елементи

**{**

**for(**int p **=** 0**;** p **<** i**;** **++**p**)**

**{**

sum **+=** **(**L**[**i**][**p**]** **\*** L**[**i**][**p**]);**

**}**

L**[**i**][**i**]** **=** sqrt**(**matrixN**[**i**][**i**]** **-** sum**);**

**continue;**

**}**

**if(**i **>** j**)**//заповнюємо всі інші елементи, які нижче за головну діагональ

**{**

**for(**int p **=** 0**;** p **<** j**;** **++**p**)**

**{**

sum **+=** **(**L**[**i**][**p**]** **\*** L**[**j**][**p**]);**

**}**

L**[**i**][**j**]** **=** **(**matrixN**[**i**][**j**]** **-** sum**)** **/** L**[**j**][**j**];**

**}**

**}**

**}**

cout **<<** "\nL matrix" **<<** endl**;**

cout **<<** endl**;**

**for(**int i **=** 0 **;** i **<** column**;** **++**i**)**

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

cout **<<** setw**(**10**)** **<<** L**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**// транспонування матриці L

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

LT**[**i**][**j**]** **=** L**[**j**][**i**];**

**}**

**}**

cout **<<** endl**;**

cout **<<** "\nLT matrix" **<<** endl **<<** endl**;**

**for(**int i **=** 0 **;** i **<** column**;** **++**i**)**// output matrix LT

**{**

**for(**int j **=** 0**;** j **<** column**;** **++**j**)**

**{**

cout **<<** setw**(**10**)** **<<** LT**[**i**][**j**]** **<<** " "**;**

**}**

cout **<<** endl**;**

**}**

vector **<**double**>** Y**(**column**,** 0.**);**

double sum\_element **=** 0.**;**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**

**{**

sum\_element **=** 0.**;**

**for(**int j **=** 0**;** j **<** i**;** **++**j**)**

**{**

sum\_element **+=** **(**Y**[**j**]** **\*** L**[**i**][**j**]);**

**}**

Y**[**i**]** **=** **(**matrixN**[**i**][**column**]** **-** sum\_element**)/**L**[**i**][**i**];**

**}**

cout **<<** endl**;**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**

**{**

cout **<<** "Y" **<<** **(**i**+**1**)** **<<**" = " **<<** Y**[**i**]** **<<** endl**;**

**}**

vector**<**double**>** roots**(**column**,** 0.**);**

cout **<<** endl **<<** endl**;**

**for(**int i **=** column **-** 1**;** i **>=** 0**;** **--**i**)**

**{**

double sum\_element **=** 0.**;**

**for(**int j **=** column **-** 1**;** j **>** i**;** **--**j**)**

**{**

sum\_element **+=** **(**roots**[**j**]** **\*** LT**[**i**][**j**]);**

**}**

roots**[**i**]** **=** **(**Y**[**i**]** **-** sum\_element**)** **/** LT**[**i**][**i**];**

**}**

**for(**int i **=** 0**;** i **<** column**;** **++**i**)**

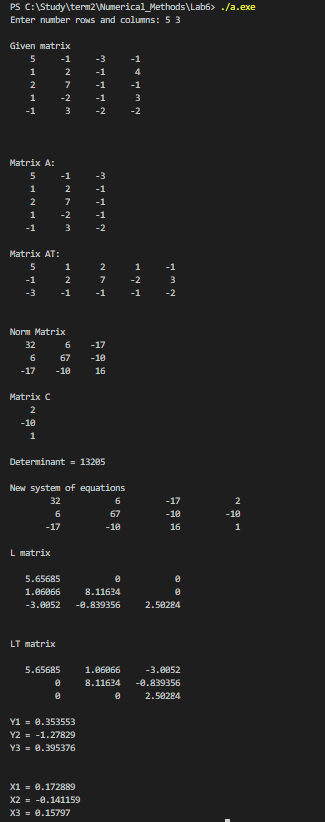
**{**

cout **<<** "X" **<<** **(**i **+** 1**)** **<<** " = " **<<** roots**[**i**]** **<<** endl**;**

**}**

**}**

**Протокол роботи**



**Висновки**

Під час виконання лабораторної роботи №6 я ознайомився на практиці з методами розв’язування перевизначених систем лінійних алгебраїчних рівнянь.