Чернівецький національний університет імені Юрія Федьковича

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра математичних проблем управління і кібернетики

Звіт

про виконання ***лабораторної роботи №2***

«***Точне та наближене розв’язування систем лінійних алгебраїчних***

***рівнянь***»

з дисципліни

“Чисельні методи”

Виконала команда «Індивідуалки»

(студенти 341 групи

Войтоловський Веніамін Ілліч,

Плакош Михайло Володимирович,

Гавучак Назар Вікторович)

Перевірила: ................ас. Філіпчук Ольга Ігорівна

Оцінка:

Дата захисту

Чернівці 2021

**Мета:** ознайомлення студентів з основними поняттями, точними та наближеними методами розв’язування СЛАР; набуття практичних навичок розв’язання таких задач (у тому числі - з використанням комп’ютера).

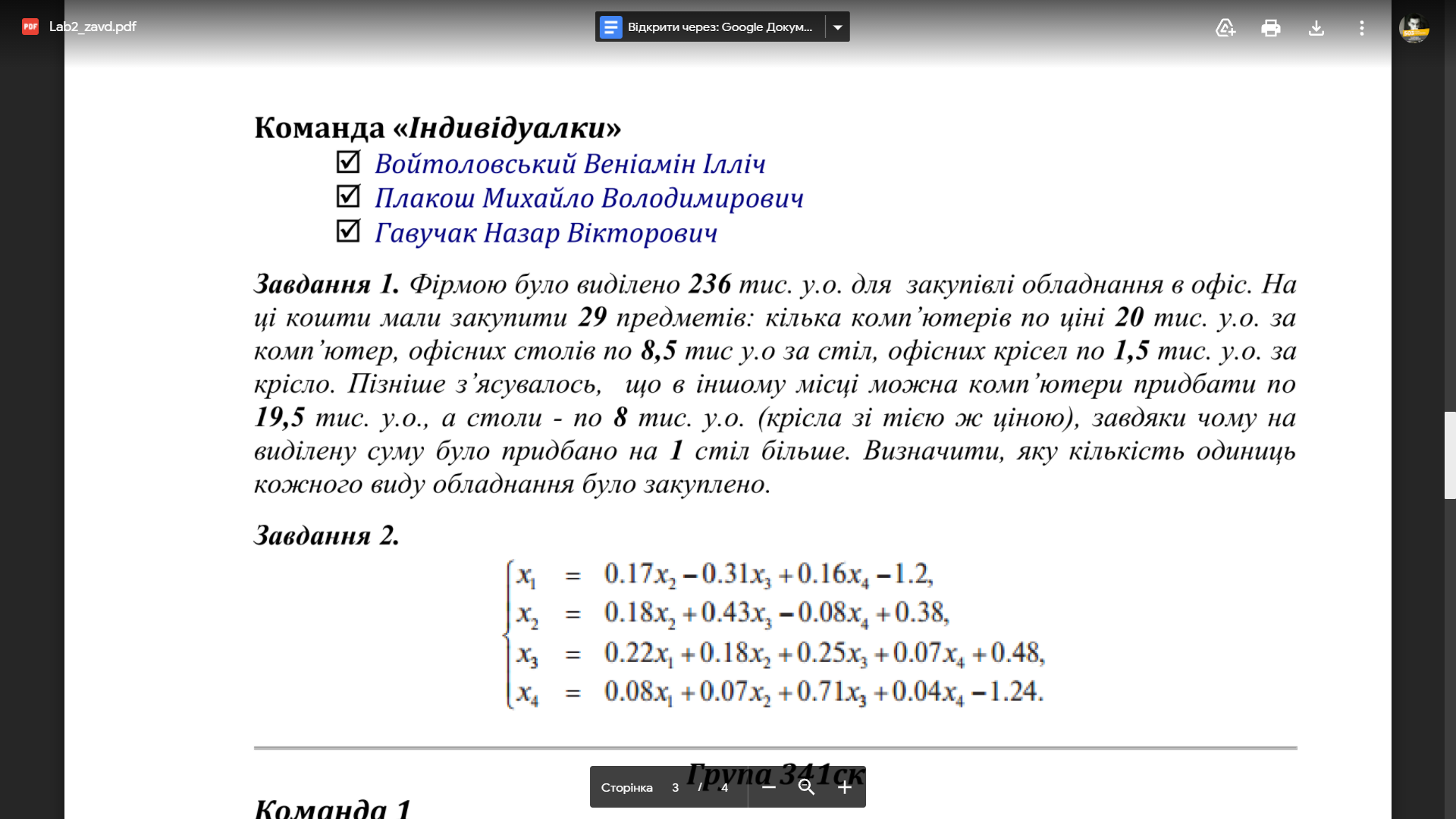
***Завдання***

***Завдання 1.*** Скласти математичну модель задачі (завдання 1). Розв’язати отриману СЛАР точними методами:

* ***методом Крамера;***
* ***методом Гаусса з вибором головного елемента у стовпці;***
* ***матричним методом*** (обернену матрицю знайти методом Гаусса);
* ***методом LU-розкладу***

***Завдання 2.*** Знайти наближений розв’язок заданої СЛАР (завдання 2) з точністю до :

* ***методом простої ітерації;***
* ***методом Зейделя.***



***Розв’язання:***

1. А) Математична модель задачі.

Нехай – запланована спочатку кількість комп’ютерів, яку мали придбати, відповідно – запланована спочатку кількість офісних столів, – кількість офісних крісел. Відомо, що на закупівлю цього обладнання було виділено 236 000 грн., і при цінах відповідно 20 000, 8500 і 1500 грн. предметів у сумі мало бути 29. Отже, І і ІІ рівняння системи:

Відомо також, що за нижчими цінами за ту ж суму вдалося придбати на 1 стіл більше. Маємо ІІІ рівняння системи:

Дещо спростивши останнє рівняння, отримуємо СЛАР, яка і є математичною моделлю даної задачі:

Оскільки – кількості одиниць придбаного обладнання, то додатково накладаються умови

Б) Розв’язання отриманої СЛАР точними методами.

1) Метод Крамера.

Математична модель:

Система має вигляд:

Матриця:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 1 | 1 | 1 | |  |  |  | |  |

Вектор: (236000,29,228000)

1.Знайдемо спільний визначник

= 5750000

Так як визначник основної матриці не дорівнює нулю, то знайдемо інші визначники:

Перший визначник виходить заміною першого стовпчика на стовпець з правої частини системи рівнянь:

= 40250000

Другий визначник виходить заміною другого стовпчика на стовпець з правої частини системи рівнянь:

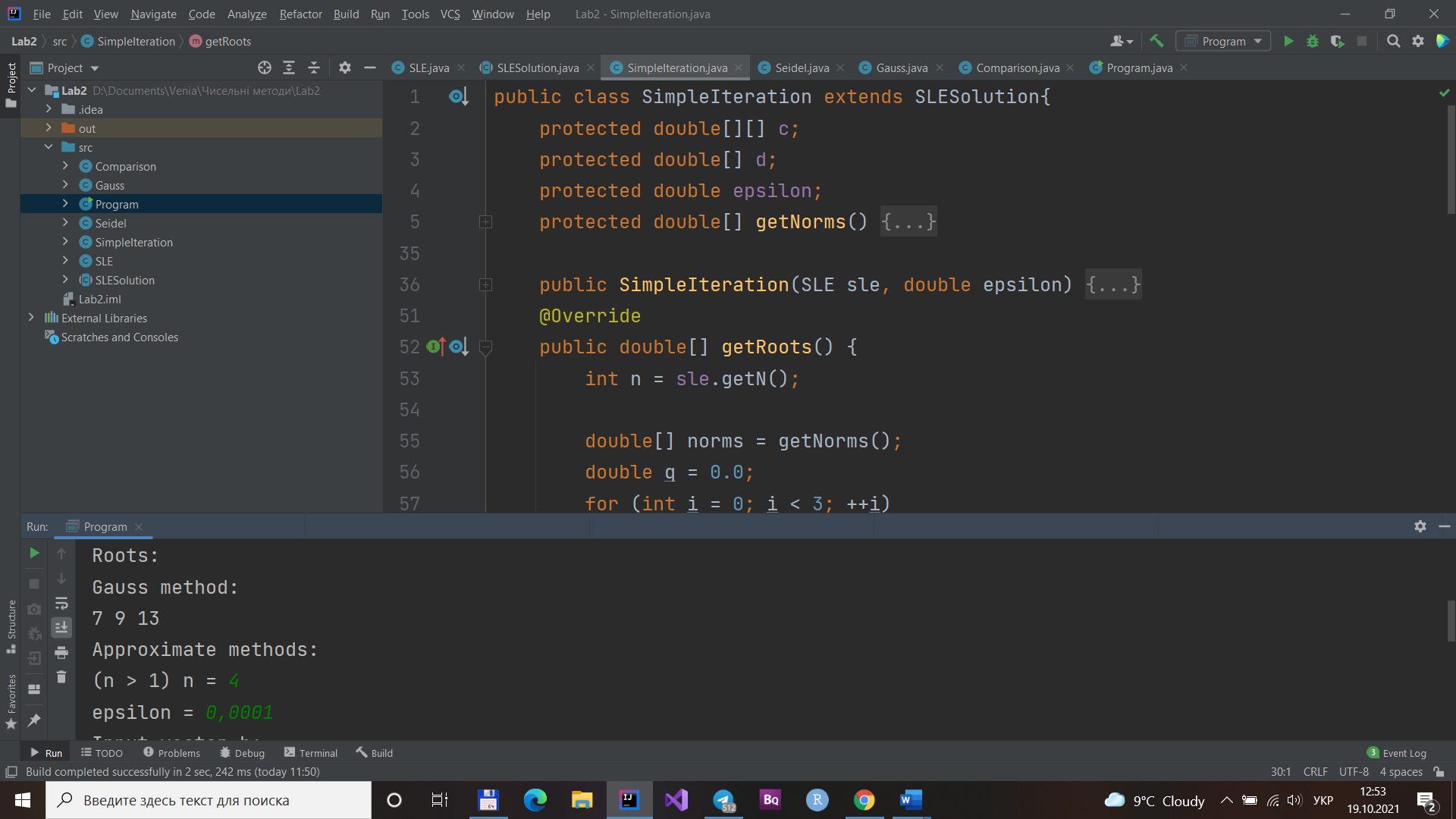
= 51750000

Третій визначник виходить заміною третього стовпчика на стовпець з правої частини системи рівнянь:

= 74750000

2) Метод Гаусса з вибором головного елемента у стовпці.

Даний метод реалізовували програмно. Отримали розв’язок:



Код методу Гаусса є в додатку.

3) Матричний метод.

**//РОЗВ’ЯЗАТИ**

4) Метод LU-розкладу.

**//РОЗВ’ЯЗАТИ**

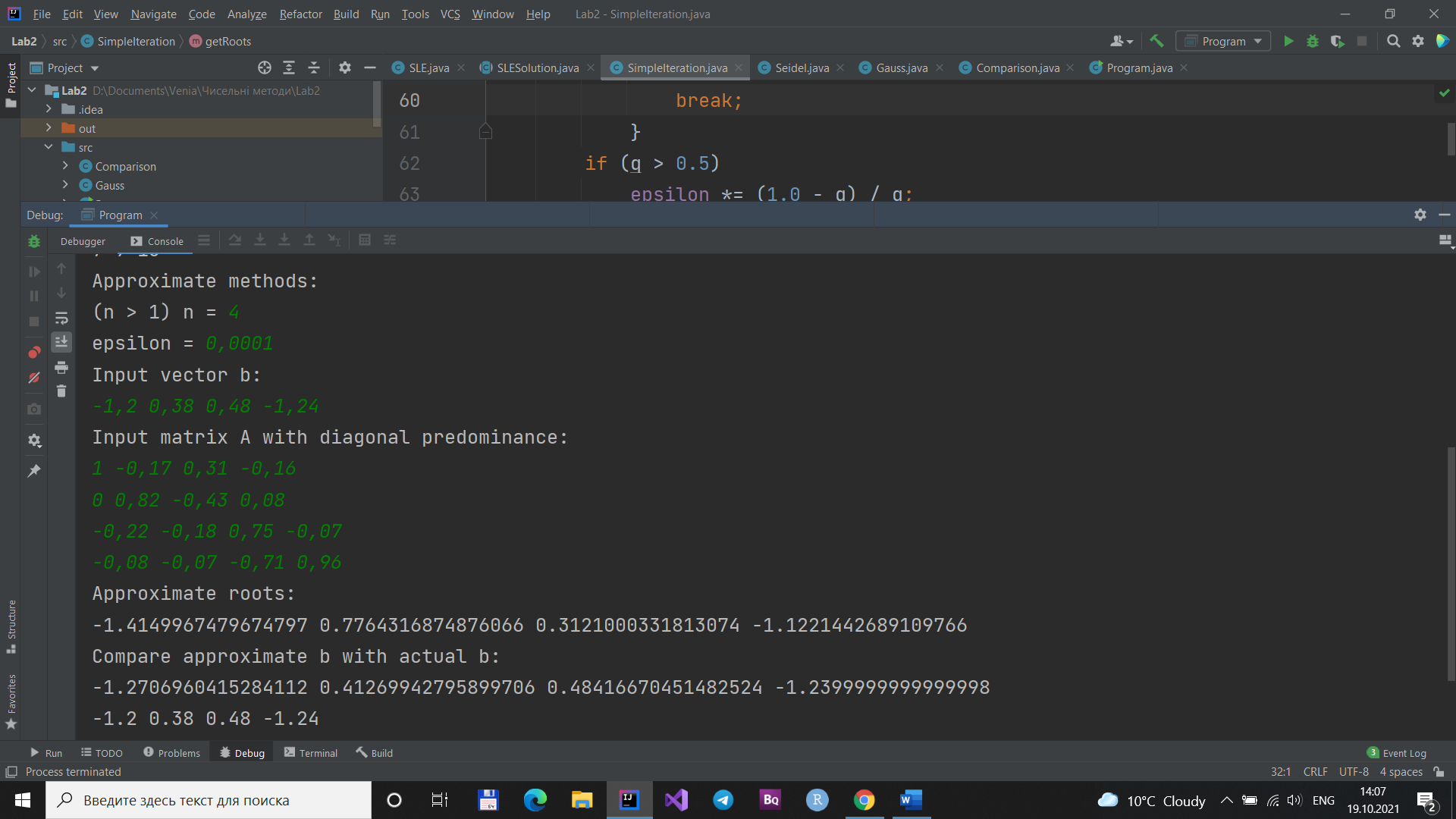
2. 1) Метод простої ітерації.

Наша система не знаходиться ні у вигляді , ні у вигляді . Перенесемо всі ікси в ліву сторону, зведемо подібні члени. Отримаємо матрицю:

.

Очевидно, що вона має діагональне переважання:

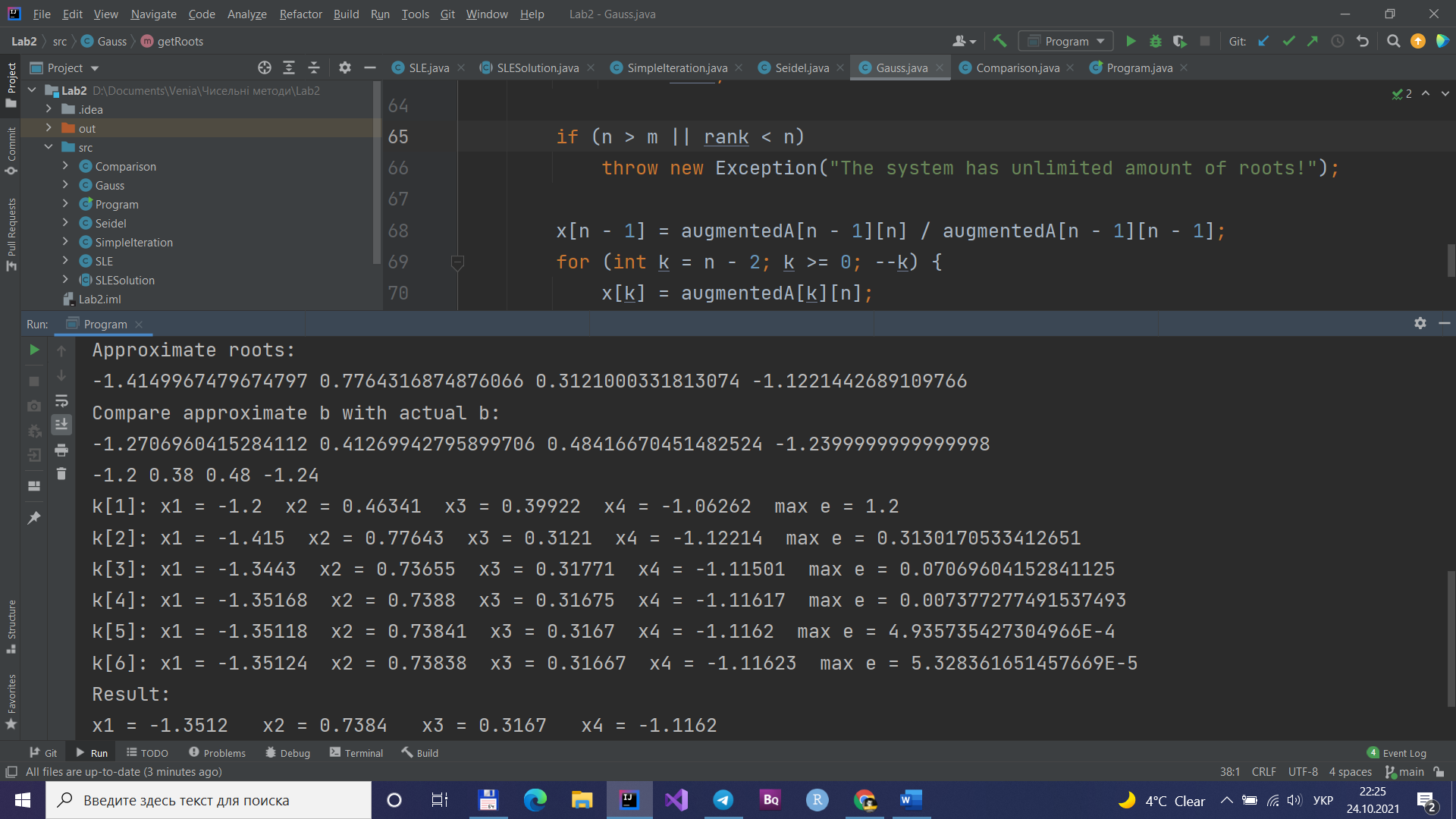
Введемо цю матрицю, а також вектор , а вона вже зведе матрицю до нормального вигляду та розв’яже її:



Бачимо наближені розв’язки, а також, підставивши їх у ліву частину, порівнюємо її з правою (ліва вище, права – нижче).

2) Метод Зейделя.

Даний метод реалізовували програмно. Отримали розв’язок за 6 ітерацій:



На даному рисунку ми можемо побачити наближення на кожній ітерації, а також максимум різниці між наближеними коренями на к-ій і (к-1)-ій ітераціях.

**Висновок:** ми ознайомилися з основними поняттями, точними та наближеними методами розв’язування СЛАР; набули практичні навички розв’язання таких задач (у тому числі - з використанням комп’ютера).

***Додаток.***

Код класів:

Система лінійних алгебраїчних рівнянь:

public class SLE {

private double[][] a;

private double[] b;

private int n;

private int m;

public double[][] getAugmentedMatrix() {

double[][] augmentedA = new double[m][n + 1];

for(int i = 0; i < m; ++i) {

for (int j = 0; j < n; ++j)

augmentedA[i][j] = a[i][j];

augmentedA[i][n] = b[i];

}

return augmentedA;

}

public SLE(double[][] a, double[] b) {

this.a = a;

this.b = b;

m = a.length;

n = a[0].length;

}

public int getN() {

return n;

}

public int getM() {

return m;

}

public double[][] getA() {

return a;

}

public double[] getB() {

return b;

}

}

Абстрактний клас для розв’язування СЛАР:

public abstract class SLESolution {

protected SLE sle;

public SLESolution(SLE sle) {

this.sle = sle;

}

public abstract double[] getRoots() throws Exception;

}

Метод Гаусса:

public class Gauss extends SLESolution{

private int findMaxDiagCoef(int pos, double[][] matrix) {

int max = pos;

for (int i = pos + 1; i < sle.getM(); ++i)

if(Math.abs(matrix[i][pos]) > Math.abs(matrix[max][pos]))

max = i;

return max;

}

public Gauss(SLE sle) {

super(sle);

}

@Override

public double[] getRoots() throws Exception{

int m = sle.getM();

int n = sle.getN();

double[][] augmentedA = sle.getAugmentedMatrix();

double[] x = new double[n];

for (int k = 0; k < Math.min(m, n); ++k) {

int maxPos = findMaxDiagCoef(k, augmentedA);

if (maxPos != k) {

double[] maxRow = augmentedA[maxPos];

augmentedA[maxPos] = augmentedA[k];

augmentedA[k] = maxRow;

}

double main = augmentedA[k][k];

if(!Comparison.areEqual(main, 0.0)) {

for (int i = k + 1; i < m; ++i) {

double mik = -augmentedA[i][k] / main;

for (int j = k + 1; j < n + 1; ++j)

augmentedA[i][j] += mik \* augmentedA[k][j];

augmentedA[i][k] = 0.0;

}

for (int i = k; i < n + 1; ++i)

augmentedA[k][i] /= main;

}

}

boolean isCompatible = true;

for (int i = m - 1; i > 0; --i) {

boolean isZeroRow = true;

for (int j = 0; j < n; ++j)

if (!Comparison.areEqual(augmentedA[i][j], 0.0)) {

isZeroRow = false;

break;

}

if (isZeroRow && !Comparison.areEqual(augmentedA[i][n], 0.0)) {

isCompatible = false;

break;

}

}

if (!isCompatible)

throw new Exception("The system is not compatible!");

int rank = 0;

for (int i = 0; i < Math.min(m, n); ++i)

if (augmentedA[i][i] != 0.0)

++rank;

if (n > m || rank < n)

throw new Exception("The system has unlimited amount of roots!");

x[n - 1] = augmentedA[n - 1][n] / augmentedA[n - 1][n - 1];

for (int k = n - 2; k >= 0; --k) {

x[k] = augmentedA[k][n];

for (int j = k + 1; j < n; ++j)

x[k] -= augmentedA[k][j] \* x[j];

x[k] /= augmentedA[k][k];

}

return x;

}

}

Метод простої ітерації:

public class SimpleIteration extends SLESolution{

protected double[][] c;

protected double[] d;

protected double epsilon;

protected double[] getNorms() {

double[] norms = new double[3];

norms[0] = 0.0;

double sum;

for (int i = 0; i < sle.getN(); ++i) {

sum = 0.0;

for (int j = 0; j < sle.getN(); ++j)

sum += Math.abs(c[i][j]);

if (sum > norms[0])

norms[0] = sum;

}

norms[1] = 0.0;

for (int j = 0; j < sle.getN(); ++j) {

sum = 0.0;

for (int i = 0; i < sle.getN(); ++i)

sum += Math.abs(c[i][j]);

if (sum > norms[1])

norms[1] = sum;

}

norms[2] = 0.0;

for (int i = 0; i < sle.getN(); ++i)

for (int j = 0; j < sle.getN(); ++j)

norms[2] += c[i][j] \* c[i][j];

norms[2] = Math.sqrt(norms[2]);

return norms;

}

public SimpleIteration(SLE sle, double epsilon) {

super(sle);

this.epsilon = epsilon;

double[][] a = sle.getA();

double[] b = sle.getB();

c = new double[sle.getN()][sle.getN()];

d = new double[sle.getN()];

for (int i = 0; i < sle.getN(); ++i) {

d[i] = b[i] / a[i][i];

for (int j = 0; j < sle.getN(); ++j)

c[i][j] = (i == j) ? 0 : - a[i][j] / a[i][i];

}

}

@Override

public double[] getRoots() {

int n = sle.getN();

double[] norms = getNorms();

double q = 0.0;

for (int i = 0; i < 3; ++i)

if (norms[i] < 1.0) {

q = norms[i];

break;

}

if (q > 0.5)

epsilon \*= (1.0 - q) / q;

double[] x = new double[n];

double[] prevX = new double[n];

for (int i = 0; i < n; ++i)

prevX[i] = 0.0;

double max;

do {

max = 0.0;

for (int i = 0; i < n; ++i) {

x[i] = d[i];

for (int j = 0; j < n; ++j)

x[i] += c[i][j] \* prevX[j];

double diff = Math.abs(x[i] - prevX[i]);

if (diff > max)

max = diff;

prevX = x;

}

} while (max > epsilon);

return x;

}

public boolean isDiagonalPredominant() {

double[][] a = sle.getA();

for (int i = 0; i < sle.getN(); ++i) {

double sum = 0;

for (int j = 0; j < sle.getN() && j != i; ++j)

sum += Math.abs(a[i][j]);

if (Math.abs(a[i][i]) <= sum)

return false;

}

return true;

}

public double[] Substitution(double[] x) {

double[][] a = sle.getA();

double[] approxB = new double[sle.getN()];

for (int i = 0; i < sle.getN(); ++i) {

approxB[i] = 0.0;

for (int j = 0; j < sle.getN(); ++j)

approxB[i] += a[i][j] \* x[j];

}

return approxB;

}

}

Метод Зейделя:

public class Seidel{

void SeidelMethod(double[][] A) {

double x = 0, y = 0, z = 0, d = 0, x1 = 0, y1 = 0, z1 = 0, d1 = 0;

double epsilon = 0.0001;

double termination = 2 \* epsilon;

for (int i = 1; termination >= epsilon; i++)

{

x1 = (A[0][1] \* y1 + A[0][2] \* z1 + A[0][3] \* d1 + A[0][4]);

y1 = (A[1][2] \* z1 + A[1][3] \* d1 + A[1][4]) / A[1][1];

z1 = (A[2][0] \* x1 + A[2][1] \* y1 + A[2][3] \* d1 + A[2][4] )/ A[2][2];

d1 = (A[3][0] \* x1 + A[3][1] \* y1 + A[3][2] \* z1 + A[3][4]) / A[3][3];

termination = Math.max(Math.max(Math.abs(x1 - x), Math.abs(y1 - y)), Math.max(Math.abs(z1 - z), Math.abs(d1 - d)));

x = x1; y = y1; z = z1; d = d1;

System.out.println(

"k[" + i + "]: " +

"x1 = " + Math.round(x \* 100000.0) / 100000.0 +

" x2 = " + Math.round(y \* 100000.0) / 100000.0 +

" x3 = " + Math.round(z \* 100000.0) / 100000.0 +

" x4 = " + Math.round(d \* 100000.0) / 100000.0 +

" max e = " + termination);

}

System.out.println("Result:");

System.out.println(

"x1 = " + Math.round(x \* 10000.0) / 10000.0 +

" x2 = " + Math.round(y \* 10000.0) / 10000.0 +

" x3 = " + Math.round(z \* 10000.0) / 10000.0 +

" x4 = " + Math.round(d \* 10000.0) / 10000.0);

}

}

Основна програма:

import java.util.Scanner;

public class Program {

private static void printVector(double[] vec) {

for (double v : vec) System.out.print(v + " ");

System.out.println();

}

private static double[][] inputMatrix(int m, int n, Scanner in) {

double[][] a = new double[m][n];

for (int i = 0; i < m; ++i)

for (int j = 0; j < n; ++j)

a[i][j] = in.nextDouble();

return a;

}

private static double[] inputVector(int m, Scanner in) {

double[] b = new double[m];

for (int i = 0; i < m; ++i)

b[i] = in.nextDouble();

return b;

}

public static void main(String[] args) {

System.out.println("Accurate methods:");

int m, n;

Scanner in = new Scanner(System.in);

do {

System.out.print("(m > 1 && n > 1) m, n = ");

m = in.nextInt();

n = in.nextInt();

} while (m <= 1 || n <= 1);

System.out.println("Input matrix A:");

double[][] a = inputMatrix(m, n, in);

System.out.println("Input vector b:");

double[] b = inputVector(m, in);

Gauss gauss = new Gauss(new SLE(a, b));

double[] x;

try {

x = gauss.getRoots();

System.out.println("Roots:");

System.out.println("Gauss method:");

for (int i = 0; i < n; ++i) {

System.out.format("%.0f ", x[i]);

}

System.out.println();

}

catch (Exception exception) {

System.out.println(exception.getMessage());

}

//If you write accurate methods, test them here.

//For an accurate method create a separate class

//which extents SLESolution.

System.out.println("Approximate methods:");

do {

System.out.print("(n > 1) n = ");

n = in.nextInt();

} while (n <= 1);

SimpleIteration simpleIteration;

System.out.print("epsilon = ");

double epsilon = in.nextDouble();

System.out.println("Input vector b:");

b = inputVector(n, in);

do {

System.out.println("Input matrix A with diagonal predominance:");

a = inputMatrix(n, n, in);

simpleIteration = new SimpleIteration(new SLE(a, b), epsilon);

} while (!simpleIteration.isDiagonalPredominant());

System.out.println("Approximate roots:");

x = simpleIteration.getRoots();

printVector(x);

System.out.println("Compare approximate b with actual b:");

printVector(simpleIteration.Substitution(x));

printVector(b);

//Test Seidel's method here.

double[][] A = {

{ 0, 0.17, -0.31, 0.16, -1.2 },

{ 0, 0.82, 0.43, -0.08, 0.38 },

{ 0.22, 0.18, 0.75, 0.07, 0.48 },

{ 0.08, 0.07, 0.71, 0.96, -1.24 },

};

Seidel seidel = new Seidel();;

seidel.SeidelMethod(A);

}

}

Утиліта для порівнювання дійсних чисел:

public class Comparison {

public static boolean areEqual(double d1, double d2) {

double eps = 1e-9;

if(Math.abs(d2 - d1) < eps)

return true;

return false;

}

}