Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій

ЗВІТ

Про виконання лабораторної роботи № 6

«Розв'язування перевизначених систем лінійних алгебраїчних рівнянь»

з дисципліни «Чисельні методи»

Лектор:
доцент кафедри ПЗ
Мельник Н.Б.

Виконав:
студ. групи ПЗ-15
Марущак А. С.

Прийняв:
асистент кафедри ПЗ
Гарматій Г.Ю
«___» ______ 2022 р.
∑ = ____ ___________

Львів – 2022

Тема роботи: Розв'язування перевизначених систем лінійних алгебраїчних рівнянь

Мета роботи: ознайомлення на практиці з методами розв’язування перевизначених систем лінійних алгебраїчних рівнянь.

Теоретичні відомості
Метод найменших квадратів для розв’язування перевизначених СЛАР

Розглянемо систему лінійних алгебраїчних рівнянь, у якій кількість рівнянь є більшою за кількість невідомих

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m = b_2, \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m = b_n, \end{cases} \text{де } \boldsymbol{n > m}$$

У загальному випадку ця система рівнянь є несумісною. Якщо із даної системи вибрати m рівнянь та розв’язати їх, то отриманий розв’язок не буде задовольняти всі рівняння цієї системи. Тому вчинимо інакше: знайдемо розв’язок системи $x_1, x_2, \dots, x_m$ наближено, але щоб він задовольняв усі рівняння системи, а саме

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m - b_1 = \varepsilon_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m - b_2 = \varepsilon_2, \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m - b_n = \varepsilon_n, \end{cases}$$

Розв’язок системи будемо знаходити з використанням умови мінімізації суми квадратів відхилень, тобто з умов мінімізації функції

$$S(x_1, x_2, \dots, x_m) = \sum_{j=1}^{m} \varepsilon_i^2$$

і вимагатимемо, щоб виконувалась умова

$$\sum_{j=1}^{m} \varepsilon_i^2 \to min$$

Проведемо деякі перетворення над системою, використовуючи цю умову. Розглянемо функцію

$$S(x_1, x_2, \dots, x_m) = \sum_{i=1}^{n} (\sum_{j=1}^{m} a_{ij}x_j - b_j)^2$$

Необхідною умовою мінімуму функції від багатьох змінних є рівність нулеві її частинних похідних. Використаємо цей факт і продиференціюємо цю функцію за змінними x_i (i =1,m). У результаті отримаємо

$$\frac{\partial S}{\partial x_k} = 2\sum_{i=1}^{n} a_{ik}\left(\sum_{j=1}^{m} a_{ij}x_j - b_j\right), \quad k = 1, m$$

Прирівнявши вирази до нуля, отримаємо нормальну систему рівнянь

$$\sum_{i=1}^{n} a_{\boldsymbol{ik}} \left(\sum_{j=1}^{m} a_{\boldsymbol{ij}} x_{\boldsymbol{j}} - b_{\boldsymbol{j}} \right) = 0, \qquad k = 1, m$$

в якій кількість рівнянь системи дорівнює кількості невідомих. Нормальні системи лінійних алгебраїчних рівнянь характеризуються тим, що матриці їх коефіцієнтів завжди є симетричними, а діагональні елементи - додатніми. Цю систему розв'язують довільними прямими або ітераційними методами. Якщо матриця коефіцієнтів системи рівнянь є додатньо визначеною (визначник матриці є більшим за нуль), то рекомендують для її розв’язування використовувати метод квадратного кореня. Запишемо систему лінійних алгебраїчних рівнянь у матричному вигляді

$$AX = B$$

$$\text{A}=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \\ \beta_n \end{bmatrix}, X = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_n \end{bmatrix}$$

де A - матриця коефіцієнтів системи розмірності m×n, X – матриця-стовпець невідомих розмірності m ×1, B - матриця-стовпець вільних членів системи розмірності m×1.

Це матричне рівняння помножимо на транспоновану матрицю A^{T} до матриці A. У результаті отримаємо матричне рівняння

$$NX = C$$

де N – матриця коефіцієнтів нормальної системи

$$N = \text{A}^{\text{T}}\text{A}$$

C -стовпець вільних членів

$$C = \text{A}^{\text{T}}B$$

Розв’язавши нормальну систему лінійних алгебраїчних рівнянь, отримаємо її точний розв’язок (якщо використано прямі методи) або наближений розв’язок (якщо використано ітераційні методи). Отриманий розв’язок буде наближеним для СЛАР.

Індивідуальне завдання

1. Ознайомитися з теоретичним матеріалом.
2. Скласти програму розв’язування перевизначеної системи лінійних алгебраїчних рівнянь методом найменших квадратів. Отриману відповідну систему розв’язати методом квадратного кореня.

Варіант завдання

$$\begin{cases} x_1 + x_2 - 4x_3 = -1 \\ 4x_1 - x_2 - 2x_3 = 7 \\ -x_1 - 5x_3 = -5 \\ 2x_1 - x_2 + 3x_3 = -7 \\ -x_1 + 3x_2 + 2x_3 = 7 \end{cases}$$

Хід роботи

Проведемо деякі обчилення вручну, щоб потім мати змогу порівняти результат виконання програми з дійсним.

Випишемо матрицю коефіцієнтів системи А, і вектор-стовпець вільних членів В:

$$A = \begin{pmatrix} 1 & 1 & -4 \\ 4 & -1 & -2 \\ -1 & 0 & -5 \\ 2 & -1 & 3 \\ -1 & 3 & 2 \end{pmatrix}; B = \begin{pmatrix} -1 \\ 7 \\ -5 \\ -7 \\ 7 \end{pmatrix}$$

Тоді наше рівняння можна записати у вигляді

$$AX = B,$$

де $X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ – вектор-стовпець невідомих. Знайдемо матрицю коефіцієнтів відповідної нормальної системи рівнянь

$$A^T A = \begin{pmatrix} 1 & 4 & -1 & 2 & -1 \\ 1 & -1 & 0 & -1 & 3 \\ -4 & -2 & -5 & 3 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 & -4 \\ 4 & -1 & -2 \\ -1 & 0 & -5 \\ 2 & -1 & 3 \\ -1 & 3 & 2 \end{pmatrix} = \begin{pmatrix} 23 & -8 & -3 \\ -8 & 12 & 1 \\ -3 & 1 & 58 \end{pmatrix} = N;$$

та матрицю-стовпець вільних членів

$$A^T B = \begin{pmatrix} 1 & 4 & -1 & 2 & -1 \\ 1 & -1 & 0 & -1 & 3 \\ -4 & -2 & -5 & 3 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ 7 \\ -5 \\ -7 \\ 7 \end{pmatrix} = \begin{pmatrix} 11 \\ 20 \\ 8 \end{pmatrix} = C;$$

Тоді систему можна записати у вигляді

$$NX = C$$

Оскільки $\det N = 12213 > 0$, то подану систему розв'яжемо методом квадратних коренів.
Нехай $N = LL^T$, тоді

$$l_{11} = \sqrt{n_{11}} = \sqrt{23};\ l_{12} = 0;\ l_{13} = 0;$$

$$l_{21} = \frac{n_{21}}{l_{11}} = -\frac{8}{\sqrt{23}}; l_{22} = \sqrt{n_{22} - l_{21}^2} = \sqrt{\frac{212}{23}};\ \ l_{23} = 0;$$

$$l_{31} = \frac{n_{31}}{l_{11}} = -\frac{3}{\sqrt{23}}; l_{32} = \frac{n_{32} - l_{31} \cdot l_{21}}{l_{22}} = -\frac{1}{23} \cdot \sqrt{\frac{23}{212}};$$

$$l_{33} = \sqrt{n_{33} - l_{31}^2 - l_{32}^2} = \frac{3\sqrt{71921}}{106};$$

$$L = \begin{pmatrix} \sqrt{23} & 0 & 0 \\ -\frac{8}{\sqrt{23}} & \sqrt{\frac{212}{23}} & 0 \\ -\frac{3}{\sqrt{23}} & -\frac{1}{23} \cdot \sqrt{\frac{23}{212}} & \frac{3\sqrt{71921}}{106} \end{pmatrix}$$

Перевіримо:

$$L \cdot L^T = \begin{pmatrix} \sqrt{23} & 0 & 0 \\ -\frac{8}{\sqrt{23}} & \sqrt{\frac{212}{23}} & 0 \\ -\frac{3}{\sqrt{23}} & -\frac{1}{23} \cdot \sqrt{\frac{23}{212}} & \frac{3\sqrt{71921}}{106} \end{pmatrix} \begin{pmatrix} \sqrt{23} & -\frac{8}{\sqrt{23}} & -\frac{3}{\sqrt{23}} \\ 0 & \sqrt{\frac{212}{23}} & -\frac{1}{23} \cdot \sqrt{\frac{23}{212}} \\ 0 & 0 & \frac{3\sqrt{71921}}{106} \end{pmatrix} =$$

$$\begin{pmatrix} 23 & -8 & -3 \\ -8 & 12 & 1 \\ -3 & 1 & 58 \end{pmatrix} = N;$$

Отже, розклад було здійснено правильно.

Тоді подальший розв'язок аналогічний до розв'язання системи методом LU-розкладу, де замість U – матриця L^T. Маємо $L^T X = Y$, тоді

$$L \cdot Y = C,$$

або

$$
\begin{pmatrix}
\sqrt{23} & 0 & 0 \\
-\frac{8}{\sqrt{23}} & \sqrt{\frac{212}{23}} & 0 \\
-\frac{3}{\sqrt{23}} & -\frac{1}{23}\cdot\sqrt{\frac{23}{212}} & \frac{3\sqrt{71921}}{106}
\end{pmatrix}
\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 11 \\ 20 \\ 8 \end{pmatrix}
$$

Звідки дістаємо, що $y_1 = \frac{11}{\sqrt{23}}; y_2 = \frac{274}{\sqrt{1219}}; y_3 = \frac{1012}{3\sqrt{71921}}$. Тоді, з рівняння $L^T X = Y$, яке записується у вигляді

$$
\begin{pmatrix}
\sqrt{23} & -\frac{8}{\sqrt{23}} & -\frac{3}{\sqrt{23}} \\
0 & \sqrt{\frac{212}{23}} & -\frac{1}{23}\cdot\sqrt{\frac{23}{212}} \\
0 & 0 & \frac{3\sqrt{71921}}{106}
\end{pmatrix}
\cdot
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} =
\begin{pmatrix} \frac{11}{\sqrt{23}} \\ \frac{274}{\sqrt{1219}} \\ \frac{1012}{3\sqrt{71921}} \end{pmatrix}
$$

Маємо, що $X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{743}{531} \\ \frac{1373}{531} \\ \frac{88}{531} \end{pmatrix} \approx \begin{pmatrix} 1{,}399 \\ 2{,}586 \\ 0{,}166 \end{pmatrix}$.

Наступна мета – реалізувати подані методи програмно і переконатися в достовірності результату.

Для подальшої роботи я буду використовувати структуру даних Matrix, яка відповідає матриці заданого розміру, як зрозуміло з назви. Серед її функціональних можливостей – множення та додавання матриць, пошук транспонованої, та детермінанту, розклад розширеної матриці системи на матрицю коефіцієнтів та вектор стовпець вільних членів. Покажемо це у коді:

```
public static Matrix operator*(Matrix a, Matrix b)
        {
            if (a.ColumnsCount != b.RowsCount)
                throw new ArgumentException("Cannot multiply matrices of these sizes.");
            Matrix result = new(a.RowsCount, b.ColumnsCount);

            for (int i = 0; i < result.RowsCount; i++)
                for (int j = 0; j < result.ColumnsCount; j++)
                    result[i, j] = a.Row(i) * b.Column(j);

            return result;
        }

public static Matrix operator+(Matrix a, Matrix b)
        {
            if (a.ColumnsCount != b.ColumnsCount || a.RowsCount != b.RowsCount)
                throw new ArgumentException("Cannot add matrices of different sizes.");
```

```
            Matrix result = new(a.RowsCount, b.ColumnsCount);

            for (int i = 0; i < result.RowsCount; i++)
                for (int j = 0; j < result.ColumnsCount; j++)
                    result[i, j] = a[i, j] + b[i, j];

            return result;
        }


public Matrix Transponed()
        {
            Matrix result = new(ColumnsCount, RowsCount);

            for (int i = 0; i < RowsCount; i++)
                for (int j = 0; j < ColumnsCount; j++)
                    result[j, i] = this[i, j];

            return result;
        }
public void Deconstruct(out Matrix A, out Matrix B)
        {
            A = new Matrix(RowsCount, ColumnsCount - 1);
            B = new Matrix(RowsCount, 1);

            for(int i = 0; i < RowsCount; i++)
            {
                for(int j = 0; j < ColumnsCount; j++)
                {
                    if (j == ColumnsCount - 1)
                        B[i, 0] = this[i, j];
                    else A[i, j] = this[i, j];
                }
            }
        }

public double Determinant()
        {
            if (ColumnsCount != RowsCount)
                throw new InvalidOperationException("Cannot find determinant for non-square
matrices.");
            if (ColumnsCount == 1)
                return this[0, 0];

            double result = 0;

            for (int i = 0; i < ColumnsCount; i++)
            {
                result += Math.Pow(-1, i) * this[0, i] *
this.WithoutRow(0).WithoutColumn(i).Determinant();
            }

            return result;
        }
```

І основна програма буде виглядати наступним чином:

```
using MatrixLib;

Matrix HoletskyMethod(Matrix A, Matrix B)
{
    if(A.Determinant() <= 0)
        throw new ArgumentException(nameof(A));
```

```
    Matrix X = new(B.RowsCount, 1), Y = new(B.RowsCount, 1);
    Matrix L = new(A.RowsCount, A.ColumnsCount);

    // Шукаємо матрицю L
    for (int i = 0; i < L.RowsCount; i++)
    {
        for (int j = 0; j < L.ColumnsCount; j++)
        {
            if (i == j)
            {
                double s = 0;
                for (int k = 0; k < i; k++)
                {
                    s += L[i, k] * L[i, k];
                }
                L[i, j] = Math.Sqrt(A[i, j] - s);
            }
            else if (j < i)
            {
                double s = 0;
                for (int k = 0; k < j; k++)
                {
                    s += L[i, k] * L[j, k];
                }
                L[i, j] = (1 / L[j, j]) * (A[i, j] - s);
            }
        }
    }

    Console.WriteLine($"Отримуємо матрицю L: \n{L}");
    Console.WriteLine($"Добуток L * L_T: \n{L * L.Transponed()}");

    // Шукаємо вектор-стовпець Y
    for (int i = 0; i < L.RowsCount; i++)
    {
        double s = 0;
        for (int j = 0; j < i; j++)
        {
            s += L[i, j] * Y[j,0];
        }
        Y[i, 0] = (B[i, 0] - s) / L[i, i];
    }

    // Шукаємо вектор-стовпець X
    for (int i = L.RowsCount - 1; i >= 0; i--)
    {
        double s = 0;
        for (int j = i + 1; j < L.ColumnsCount; j++)
        {
            s += L.Transponed()[i, j] * X[j, 0];
        }
        X[i, 0] = (Y[i, 0] - s) / L.Transponed()[i, i];
    }

    //Повертаємо результат
    return X;
}

//Зчитуємо матрицю з файлу
Matrix systemMatrix = Matrix.FromFile("input.txt");
//Розкладаємо розширену матрицю на матрицю коефіцієнтів і стовпець вільних членів
(Matrix A, Matrix B) = systemMatrix;

//Шукаємо нормальну матрицю системи
Matrix N = A.Transponed() * A;
```

```
Console.WriteLine($"Нормальна матриця системи: \n{N}");
//Шукаємо новий стовпець вільних членів
Matrix C = A.Transponed() * B;
Console.WriteLine($"Новий стовпець вільних членів: \n{C}");

Matrix X = HoletskyMethod(N, C);
Console.WriteLine($"Відповідь: \n{X}");
```

Тоді, з умови дістанемо розширену матрицю системи

$$\boldsymbol{A}|\boldsymbol{B} = \begin{pmatrix} 1 & 1 & -4 & -1 \\ 4 & -1 & -2 & 7 \\ -1 & 0 & -5 & -5 \\ 2 & -1 & 3 & -7 \\ -1 & 3 & 2 & 7 \end{pmatrix}$$

Подамо цю матрицю на вхід програми і подивимось на результат.

<u>Результат</u> виконання програми:

```
Нормальна матриця системи:
  23,000   -8,000   -3,000
  -8,000   12,000    1,000
  -3,000    1,000   58,000

Новий стовпець вільних членів:
 11,000
 20,000
  8,000

Отримуємо матрицю L:
   4,796    0,000    0,000
  -1,668    3,036    0,000
  -0,626   -0,014    7,590

Добуток L * L_T:
  23,000   -8,000   -3,000
  -8,000   12,000    1,000
  -3,000    1,000   58,000

Відповідь:
 1,399
 2,586
 0,166
```

Рис 6.1 Результат виконання програми.

<u>Аналіз результатів</u>:

Як бачимо, результати обчислень на комп’ютері повністю співпали з результатами обчислень, здійснених вручну. Також, співпали матриці N, C та L. Тому, можемо судити про достовірність як і ручних обчислень, так і програми, складеної на основі вивченого матеріалу.

Так як пряма підстановка результату в початкову систему дасть мало інформації, то аналітично доведемо, що отриманий роз’язок є точкою мінімуму для функції похибки. Запишемо похибку кожного рядка:

$$\begin{cases} x_1 + x_2 - 4x_3 + 1 = \varepsilon_1 \\ 4x_1 - x_2 - 2x_3 - 7 = \varepsilon_2 \\ -x_1 - 5x_3 + 5 = \varepsilon_3 \\ 2x_1 - x_2 + 3x_3 + 7 = \varepsilon_4 \\ -x_1 + 3x_2 + 2x_3 - 7 = \varepsilon_5 \end{cases}$$

Тоді функція похибки $E(x_1, x_2, x_3) = \sum_{i=1}^{5} \varepsilon_i^2 = (x_1 + x_2 - 4x_3 + 1)^2 + (4x_1 - x_2 - 2x_3 - 7)^2 + (-x_1 - 5x_3 + 5)^2 + (2x_1 - x_2 + 3x_3 + 7)^2 + (-x_1 + 3x_2 + 2x_3 - 7)^2 = 23x_1^2 - 16x_1x_2 - 6x_1x_3 - 22x_1 + 12x_2^2 + 2x_2x_3 - 40x_2 + 58x_3^2 - 16x_3 + 173.$

Градієнт функції $\overline{grad}(E) = \frac{\partial E}{\partial x_1}\bar{\imath} + \frac{\partial E}{\partial x_2}\bar{\jmath} + \frac{\partial E}{\partial x_3}\bar{k}$ – напрямок найшвидшого росту. Тоді очевидно, що для мінімізації функції треба рухатись в напрямку, протилежному градієнту в точці, тобто за напрямком вектора $-\overline{grad}(E)$. І запишемо наступний алгоритм:

1. Вибрати довільну точку A.
2. Знайти градієнт функції в точці A.
3. Перемістити точку A за вектором $-\overline{grad}(E) \cdot s$, де $s = const$ – коефіцієнт, який ми обираємо самі і який показує швидкість наближення.
4. Повторювати кроки 2,3 допоки відстань між попереднім і теперішнім положенням точки не стане менше за точність ε.

Знайдемо частинні похідні:

$$\frac{\partial E}{\partial x_1} = 46x_1 - 16x_2 - 6x_3 - 22$$
$$\frac{\partial E}{\partial x_2} = -16x_1 + 24x_2 + 2x_3 - 40$$
$$\frac{\partial E}{\partial x_3} = -6x_1 + 2x_2 + 116x_3 - 16$$

І тоді на основі цього алгоритму напишемо коротку програму, вибравши за початкове наближення точку з координатами (0, 0, 0):

```
double f(double x, double y, double z) => 23*x*x - 16*x*y - 6*x*z - 22*x + 12*y*y + 2*y*z - 40*y + 58*z*z - 16*z + 173;
double dfdx(double x, double y, double z) => 46 * x - 16 * y - 6 * z - 22;
double dfdy(double x, double y, double z) => -16*x + 24*y + 2*z - 40;
double dfdz(double x, double y, double z) => -6 * x + 2 * y + 116 * z - 16;
Vector grad(double x, double y, double z) => new Vector(
            dfdx(x, y, z),
            dfdy(x, y, z),
            dfdz(x, y, z)
        );

const double step = 0.01;
Point A = new(0, 0, 0), A_prev;
```

```
do
{
    A_prev = (A.Clone() as Point)!;
    double x = A.X;
    double y = A.Y;
    double z = A.Z;

    A = A_prev.MovedBy(-grad(x,y,z) * step);
    Console.WriteLine(A);
} while (A.DistanceTo(A_prev) > 5e-5);
```

І тоді отримуємо наступну послідовність наближень:

```
(0,220, 0,400, 0,160)
(0,412, 0,736, 0,140)
(0,569, 1,023, 0,148)
(0,700, 1,265, 0,150)
(0,809, 1,470, 0,153)
(0,901, 1,644, 0,155)
(0,979, 1,791, 0,156)
(1,045, 1,914, 0,158)
(1,100, 2,019, 0,159)
(1,146, 2,107, 0,160)
(1,186, 2,182, 0,161)
(1,219, 2,245, 0,162)
(1,247, 2,298, 0,162)
(1,271, 2,343, 0,163)
(1,291, 2,380, 0,163)
(1,308, 2,412, 0,164)
(1,322, 2,439, 0,164)
(1,334, 2,462, 0,164)
(1,344, 2,481, 0,165)
(1,353, 2,498, 0,165)
(1,360, 2,511, 0,165)
(1,366, 2,523, 0,165)
(1,371, 2,533, 0,165)
(1,376, 2,541, 0,165)
(1,379, 2,548, 0,165)
(1,382, 2,554, 0,165)
(1,385, 2,559, 0,165)
(1,387, 2,563, 0,165)
(1,389, 2,566, 0,166)
(1,391, 2,569, 0,166)
(1,392, 2,572, 0,166)
(1,393, 2,574, 0,166)
(1,394, 2,576, 0,166)
(1,395, 2,577, 0,166)
(1,396, 2,579, 0,166)
(1,396, 2,580, 0,166)
(1,397, 2,581, 0,166)
(1,397, 2,582, 0,166)
(1,397, 2,582, 0,166)
(1,398, 2,583, 0,166)
(1,398, 2,583, 0,166)
(1,398, 2,584, 0,166)
(1,398, 2,584, 0,166)
(1,398, 2,584, 0,166)
(1,399, 2,584, 0,166)
(1,399, 2,585, 0,166)
(1,399, 2,585, 0,166)
(1,399, 2,585, 0,166)
```

Рис 6.2 Результат виконання градієнтного спуску.

Як бачимо, методом градієтного спуску ми втрапили в ту саму точку, що і розв’язуючи нормальну систему. Тому це точка локального мінімуму. І щоб завершити доведення, треба довести, що цей мінімум є і глобальним.

Матриця Гесе цієї функції: $H = \begin{pmatrix} \frac{\partial^2 E}{\partial x_1^2} & \frac{\partial^2 E}{\partial x_1 \partial x_2} & \frac{\partial^2 E}{\partial x_1 \partial x_3} \\ \frac{\partial^2 E}{\partial x_2 \partial x_1} & \frac{\partial^2 E}{\partial x_2^2} & \frac{\partial^2 E}{\partial x_2 \partial x_3} \\ \frac{\partial^2 E}{\partial x_3 \partial x_1} & \frac{\partial^2 E}{\partial x_3 \partial x_2} & \frac{\partial^2 E}{\partial x_3^2} \end{pmatrix} = \begin{pmatrix} 46 & -16 & -6 \\ -16 & 24 & 2 \\ -6 & 2 & 116 \end{pmatrix}$;

Оскільки $\Delta_1 = |46| = 46 > 0$; $\Delta_2 = \begin{vmatrix} 46 & -16 \\ -16 & 24 \end{vmatrix} = 848 > 0$; $\Delta_3 \begin{vmatrix} 46 & -16 & -6 \\ -16 & 24 & 2 \\ -6 & 2 & 116 \end{vmatrix} = 97704 > 0$, то за критерієм Сильвестра ця матриця – додатньо визначена. А це означає, що функція опукла, що геометрично неможливо уявити у випадку функції 3 аргументів, проте, формально кажучи $\exists!\ (x_1^*, x_2^*, x_3^*) \in \mathbb{R}^3\ E(x_1^*, x_2^*, x_3^*) = \min E \wedge \forall (x_1, x_2, x_3) \in \mathbb{R}^3\ E(x_1, x_2, x_3) \geq \min E$. Це і означає, що функція має один мінімум і він, до того ж, глобальний. Отже, наше доведення завершене і результат підтверджений.

<u>Висновок</u>:

Виконуючи цю лабораторну роботу ми ознайомились на практиці з методами розв'язування перевизначених систем лінійних алгебраїчних рівнянь.

За допомогою цих знань ми реалізували програму, що знайшла оптимальний розв'язок наступної системи:

$$\begin{cases} x_1 + x_2 - 4x_3 = -1 \\ 4x_1 - x_2 - 2x_3 = 7 \\ -x_1 - 5x_3 = -5 \\ 2x_1 - x_2 + 3x_3 = -7 \\ -x_1 + 3x_2 + 2x_3 = 7 \end{cases}$$

Перед цим, ми провели деякі обчислення вручну, і проміжні результати обчислень повністю аналогічні до результату виконання програми.

Ми отримали вектор-стовпець дійсних чисел, що відображає найоптимальніший розв' язок цієї системи:

$$X \approx \begin{pmatrix} 1{,}399 \\ 2{,}586 \\ 0{,}166 \end{pmatrix}$$

і аналітично довели, що цей результат є вірним.