

Міністерство освіти і науки України
Національний Технічний Університет України
Київський політехнічний інститут імені Ігоря Сікорського
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №1
з дисципліни «Чисельні методи»
Тема «Розв’язання систем лінійних алгебраїчних рівнянь
(СЛАР) прямими методами»
Варіант №22

Студента 2-го курсу НН ІАТЕ гр. ТР-12

Ковальова Олександра

Перевірила: к.т.н., доц. Залевська О. В.

Мета роботи. Розробити програмну реалізацію методу Гауса, перевірити результат за допомогою вектора нев'язки.

Теоретична частина. Будемо розглядати систему

$$Ax = b, \quad (1)$$

де $A(n \times n)$ – матриця системи, b - вектор правої частини, x - вектор розв'язку.

Метод Гауса.

Метод складається з двох етапів:

- 1) прямого ходу методу (приведення системи (1) до еквівалентної системи з трикутною матрицею);
- 2) зворотного ходу (визначення невідомого вектору x).

Існує декілька варіантів методу Гауса.

Схема з вибором головного елемента полягає у наступному:

Прямий хід.

1.1) Відшукати $a_{\text{main}} = |a_{ij}|, i, j = 1 \dots n$. Нехай $a_{\text{main}} = a_{pq}$. Рядок p називається головним.

1.2) Обчислити множники $m_i = \frac{a_{iq}}{a_{pq}}, i \neq p$.

1.3) З кожного i -го неголовного рядка віднімаємо покомпонентно головний рядок, який помножено на m_i :

$$a_{ij} := a_{ij} - m_i a_{pj}, i \neq p, j = 1..n,$$

для вектора правої частини:

$$b_i := b_i - m_i b_p.$$

В результаті отримуємо матрицю, де всі елементи стовпця q , крім a_{pq} , дорівнюють нулю. Відкидаючи стовпець q та головний рядок p , і відповідний елемент b_p , отримуємо систему з матрицею $A_1((n-1) \times (n-1))$. Якщо $n-1 > 1$, покладаємо $n := n-1$, і переходимо до п.1.1, інакше переходимо до п.2.

Примітка: Елементи головного рядка та відповідного елементу b_p потрібно зберігати у окремому масиві, оскільки вони знадобляться в п.2).

Зворотний хід.

2.1) Складаємо систему, еквівалентну вихідній, що складається з головних рядків, які отримувались у п.1. Права частина складається з відповідних елементів b_p . Отримана система має трикутну матрицю. Знаходимо послідовно значення елементів x_i .

Метод квадратного кореня.

Метод використовується для розв'язання СЛАР виду (1), у яких матриця A є симетричною, тобто

$$a_{ij} = a_{ji} \quad \forall i, j$$

Метод полягає у наступному:

1) Прямий хід: факторизація $A = T'T$, де

$$T = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & t_{nn} \end{pmatrix}, \quad T' = \begin{pmatrix} t_{11} & 0 & \dots & 0 \\ t_{12} & t_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ t_{1n} & t_{2n} & \dots & t_{nn} \end{pmatrix}$$

1.1) Знаходимо елементи t_{ij} матриць-множників.

$$t_{11} = \sqrt{a_{11}}, \quad t_{1j} = \frac{a_{1j}}{t_{11}} \quad (j > 1),$$

$$t_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} t_{ki}^2} \quad (1 < i \leq n),$$

$$t_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} t_{ki} t_{kj}}{t_{ii}} \quad (i < j),$$

$$t_{ij} = 0 \quad (i > j)$$

1.2) Формуємо замість вихідної системи дві наступні системи:
 $T'y = b, \quad T'x = y.$

2) Зворотний хід.

2.1) Послідовно знаходимо:

$$y_1 = \frac{b_1}{t_{11}}, \quad y_i = \frac{b_i - \sum_{k=1}^{i-1} t_{ki} y_k}{t_{ii}} \quad (i > 1)$$

$$x_n = \frac{y_n}{t_{nn}}, \quad x_i = \frac{y_i - \sum_{k=i+1}^n t_{ik} x_k}{t_{ii}} \quad (i < n).$$

Завдання. Розв'язати систему рівнянь з кількістю значущих цифр $m = 6$. Розв'язок повинен йти через побічну діагональ.

Вивести всі проміжні результати (матриці A , що отримані в ході прямого ходу методу Гауса, матрицю зворотного ходу методу Гауса), та розв'язок системи.

Навести результат перевірки: вектор нев'язки $r = b - Ax$, де x - отриманий розв'язок. Розв'язати задану систему рівнянь за допомогою програмного забезпечення Mathcad.

Порівняти корені рівнянь, отримані у Mathcad, із власними результатами за допомогою методу середньоквадратичної похибки:

$$\delta = \sqrt{\frac{1}{n} \sum_{k=1}^n (x_k - x_{mk})^2}$$

Хід роботи

Індивідуальне завдання:

| | | |
|-----------|--|--|
| 22- 25 | $\begin{pmatrix} 5,18 + \alpha & 1,12 & 0,95 & 1,32 & 0,83 \\ 1,12 & 4,28 - \alpha & 2,12 & 0,57 & 0,91 \\ 0,95 & 2,12 & 6,13 + \alpha & 1,29 & 1,57 \\ 1,32 & 0,57 & 1,29 & 4,57 - \alpha & 1,25 \\ 0,83 & 0,91 & 1,57 & 1,25 & 5,21 + \alpha \end{pmatrix}$ $\alpha = 0,25k, k = \text{№вар} - 25 $ | $\begin{pmatrix} 6,19 + \beta \\ 3,21 \\ 4,28 - \beta \\ 6,25 \\ 4,95 + \beta \end{pmatrix}$ $\beta = 0,35k, k = \text{№вар} - 21$ |
|-----------|--|--|

Для розробки програми використовуватимемо мову програмування C# та середовище розробки JetBrains Rider.

В головному класі програми вкажемо початкові дані.

```
const int var = 22;

var a:decimal = 0.25m * Math.Abs(var - 25);
var b:decimal = 0.35m * (var - 21);

decimal[][] A =
{
    new decimal[] { 5.18m + a, 1.12m, 0.95m, 1.32m, 0.83m },
    new decimal[] { 1.12m, 4.28m - a, 2.12m, 0.57m, 0.91m },
    new decimal[] { 0.95m, 2.12m, 6.13m + a, 1.29m, 1.57m },
    new decimal[] { 1.32m, 0.57m, 1.29m, 4.57m - a, 1.25m },
    new decimal[] { 0.83m, 0.91m, 1.57m, 1.25m, 5.21m + a }
};

decimal[] B = { 6.19m + b, 3.21m, 4.28m - b, 6.25m, 4.95m + b };
```

Метод Гауса починається з прямого ходу. Прямий хід методу Гауса реалізований в класі ForwardElimination. В ньому наявний лише один метод, який називається Start. Алгоритм такий: спочатку створюємо масив, в якому будемо зберігати позиції відповідей. Вони будуть змінюватись при зміні положення стовбців. Після цього входимо в цикл, який проходиться по всім рядкам. Одночасно змінна циклу є і обмежувачем – алгоритм рухається рекурсивно в сторону зменшення матриці.

Так як діагональ побічна, то матриця буде звужуватись на один рядок зверху та один стовбець збоку (не враховуючи вектор відповідей).

Як приклад, візьмемо таку систему.

$$\begin{cases} x_1 + x_2 - x_3 = 0 \\ 2x_1 + x_2 + 2x_3 = 10 \\ x_1 - 3x_2 + x_3 = -2 \end{cases}$$

Запишемо розширену матрицю системи.

$$\left(\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 2 & 1 & 2 & 10 \\ 1 & -3 & 1 & -2 \end{array} \right)$$

Знаходимо максимум матриці – це елемент під індексом 2-2. Міняємо перший та другий рядок місцями.

$$\left(\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 1 & 1 & -1 & 0 \\ 1 & -3 & 1 & -2 \end{array} \right)$$

На цьому моменті потрібно найбільший елемент поточного рядку перемістити в кінець діапазону, який переглядається.

$$\left(\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ \hline 1 & 1 & -1 & 0 \\ \hline 1 & -3 & 1 & -2 \end{array} \right)$$

Червоними лініями виділені діапазони. Так як знаходимось в першому, то максимальний елемент повинен переміститись в комірку 1-3. Але там і так знаходиться двійка, тому ніяких маніпуляцій робити не потрібно.

На місці максимального елементу треба отримати одиницю. Для цього ділимо весь рядок на це число (у нашому випадку це 2):

$$\left(\begin{array}{ccc|c} \underline{1} & \underline{0,5} & \underline{1} & \underline{5} \\ 1 & 1 & -1 & 0 \\ 1 & -3 & 1 & -2 \end{array} \right)$$

Після цього, під правою одиницею потрібно отримати нулі. Для цього циклічно проходимося по рядкам та віднімаємо від всіх елементів рядка їх правий елемент * на цей же елемент у першому рядку. Наприклад, другий рядок. З елементом 2-1 потрібно провести такі маніпуляції: взяти елемент з першого рядку (1-1) який дорівнює 1, помножити на крайній елемент рядку з поточного діапазону (2-3, дорівнює -1). Результат дорівнює -1. Це число потрібно відняти від елемента, який оброблюємо (2-1). Тобто, отримуємо: $1 - (1 * -1) = 2$.

Формулу можна зрозуміти зі схеми:

$$\begin{array}{c}
 \begin{array}{cccc}
 1 - 1 \cdot (-1) = 2 & 1 - 0,5 \cdot (-1) = 1,5 & -1 - 1 \cdot (-1) = 0 & 0 - 5 \cdot (-1) = 5
 \end{array} \\
 \left[\begin{array}{ccc|c}
 1 & 0,5 & 1 & 5 \\
 1 & 1 & -1 & 0 \\
 1 & -3 & 1 & -2
 \end{array} \right]
 \end{array}$$

$$\left[\begin{array}{ccc|c}
 1 & 0,5 & 1 & 5 \\
 2 & 1,5 & 0 & 5 \\
 1 & -3 & 1 & -2
 \end{array} \right]$$

Як можемо побачити, під одиницею утворився 0. Такі ж дії проводимо і над 3 рядком:

$$\left[\begin{array}{ccc|c}
 1 & 0,5 & 1 & 5 \\
 2 & 1,5 & 0 & 5 \\
 0 & -3,5 & 0 & -7
 \end{array} \right]$$

Отримуємо в третьому стовбці нулі. Звужуємо діапазон:

$$\left[\begin{array}{ccc|c}
 1 & 0,5 & 1 & 5 \\
 2 & 1,5 & 0 & 5 \\
 0 & -3,5 & 0 & -7
 \end{array} \right]$$

Максимальним в матриці є елемент 2-1, тобто 2. Рядки місцями змінювати не потрібно. Але потрібно змінювати рядки. Міняємо місцями рядки 1 та 2:

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & \\ \hline 0,5 & 1 & 1 & 5 \\ 1,5 & 2 & 0 & 5 \\ -3,5 & 0 & 0 & -7 \end{array} \right]$$

Ділимо другий рядок на максимальний елемент: 2.

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & \\ \hline 0,5 & 1 & 1 & 5 \\ 0,75 & 1 & 0 & 2,5 \\ -3,5 & 0 & 0 & 7 \end{array} \right]$$

Обчислюємо третій рядок за формулою, отримуємо те саме, так як правий елемент дорівнює 0.

Звужуємо діапазон до одного елемента 3-1, тобто -3.5. Ділимо рядок на цей елемент, отримуємо 1. Отримуємо побічну одиничну діагональ.

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & \\ \hline 0,5 & 1 & 1 & 5 \\ 0,75 & 1 & 0 & 2,5 \\ 1 & 0 & 0 & 2 \end{array} \right]$$

Далі обраховуємо результати. Як можемо побачити зверху, в третьому рядку у нас знаходиться x_2 , в другому x_1 , і в третьому x_3 . Починається зворотній метод Гауса (реалізований в класі BackSubstraction). $x_2 = 2$. Підставляємо у друге рівняння:

$$2 * 0,75 + x_1 = 2,5 \rightarrow x_1 = 2,5 - 1,5 = 1$$

$$x_1 = 1.$$

$$\text{Підставляємо: } 0,5 * 2 + 1 * 1 + x_3 = 5 \rightarrow x_3 = 5 - 2 \rightarrow x_3 = 3$$

Тобто, результатами є: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$.

Звіримося з результатами програми:

```
/// !!! МАСИВ РЕЗУЛЬТАТІВ !!! ///
```

| |
|-------------|
| x1 = 000001 |
| x2 = 000002 |
| x3 = 000003 |

Результати співпадають, алгоритм працює.

Запустимо програму з даними з індивідуального завдання:

```
/// Початкова матриця, початок прямого ходу методу Гауса ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 005,93 | 001,12 | 000,95 | 001,32 | 000,83 | 006,54 | 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 |
| 001,12 | 003,53 | 002,12 | 000,57 | 000,91 | 003,21 | 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 |
| 000,95 | 002,12 | 006,88 | 001,29 | 001,57 | 003,93 | 005,93 | 001,12 | 000,83 | 001,32 | 000,95 | 006,54 |
| 001,32 | 000,57 | 001,29 | 003,82 | 001,25 | 006,25 | 001,32 | 000,57 | 001,25 | 003,82 | 001,29 | 006,25 |
| 000,83 | 000,91 | 001,57 | 001,25 | 005,96 | 005,30 | 000,83 | 000,91 | 005,96 | 001,25 | 001,57 | 005,30 |

```
/// Міняємо місцями 1 і 3 рядки ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 000,95 | 002,12 | 006,88 | 001,29 | 001,57 | 003,93 | 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 |
| 001,12 | 003,53 | 002,12 | 000,57 | 000,91 | 003,21 | 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 |
| 005,93 | 001,12 | 000,95 | 001,32 | 000,83 | 006,54 | 5,7988 | 0,8273 | 0,6132 | 1,1419 | 0 | 5,9973 |
| 001,32 | 000,57 | 001,29 | 003,82 | 001,25 | 006,25 | 001,32 | 000,57 | 001,25 | 003,82 | 001,29 | 006,25 |
| 000,83 | 000,91 | 001,57 | 001,25 | 005,96 | 005,30 | 000,83 | 000,91 | 005,96 | 001,25 | 001,57 | 005,30 |

```
/// Робимо перетворення з рядком 2 ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 005,93 | 001,12 | 000,95 | 001,32 | 000,83 | 006,54 | 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 |
| 001,12 | 003,53 | 002,12 | 000,57 | 000,91 | 003,21 | 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 |
| 000,95 | 002,12 | 006,88 | 001,29 | 001,57 | 003,93 | 5,7988 | 0,8273 | 0,6132 | 1,1419 | 0 | 5,9973 |
| 001,32 | 000,57 | 001,29 | 003,82 | 001,25 | 006,25 | 001,32 | 000,57 | 001,25 | 003,82 | 001,29 | 006,25 |
| 000,83 | 000,91 | 001,57 | 001,25 | 005,96 | 005,30 | 000,83 | 000,91 | 005,96 | 001,25 | 001,57 | 005,30 |

```
/// Змінюємо місцями 3 та 5 стовбці ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 000,95 | 002,12 | 001,57 | 001,29 | 006,88 | 003,93 | 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 |
| 001,12 | 003,53 | 000,91 | 000,57 | 002,12 | 003,21 | 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 |
| 005,93 | 001,12 | 000,83 | 001,32 | 000,95 | 006,54 | 5,7988 | 0,8273 | 0,6132 | 1,1419 | 0 | 5,9973 |
| 001,32 | 000,57 | 001,25 | 003,82 | 001,29 | 006,25 | 1,1419 | 0,1725 | 0,9556 | 3,5781 | 0 | 5,5131 |
| 000,83 | 000,91 | 005,96 | 001,25 | 001,57 | 005,30 | 000,83 | 000,91 | 005,96 | 001,25 | 001,57 | 005,30 |

```
/// Ділимо рядок 1 на локальний максимум (006,88) ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 | 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 |
| 001,12 | 003,53 | 000,91 | 000,57 | 002,12 | 003,21 | 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 |
| 005,93 | 001,12 | 000,83 | 001,32 | 000,95 | 006,54 | 5,7988 | 0,8273 | 0,6132 | 1,1419 | 0 | 5,9973 |
| 001,32 | 000,57 | 001,25 | 003,82 | 001,29 | 006,25 | 1,1419 | 0,1725 | 0,9556 | 3,5781 | 0 | 5,5131 |
| 000,83 | 000,91 | 005,96 | 001,25 | 001,57 | 005,30 | 0,6132 | 0,4262 | 5,6017 | 0,9556 | 0 | 4,4032 |

```
/// Робимо перетворення з рядком 5 ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 5,7988 | 0,8273 | 0,6132 | 1,1419 | 0 | 5,9973 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |
| 1,1419 | 0,1725 | 0,9556 | 3,5781 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,6132 | 0,4262 | 5,6017 | 0,9556 | 0 | 4,4032 | 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 |

```
/// Міняємо місцями 2 і 3 рядки ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1381 | 0,3081 | 0,2282 | 0,1875 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 5,7988 | 0,8273 | 0,6132 | 1,1419 | 0 | 5,9973 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,8273 | 2,8767 | 0,4262 | 0,1725 | 0 | 1,9990 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |
| 1,1419 | 0,1725 | 0,9556 | 3,5781 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,6132 | 0,4262 | 5,6017 | 0,9556 | 0 | 4,4032 | 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 |

```
/// Робимо перетворення з рядком 4 ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 1,1419 | 0,8273 | 0,6132 | 5,7988 | 0 | 5,9973 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,1725 | 2,8767 | 0,4262 | 0,8273 | 0 | 1,9990 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |
| 3,5781 | 0,1725 | 0,9556 | 1,1419 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 | 0,8349 | 0,3387 | 5,5369 | 0 | 0 | 3,7680 |

```
/// Змінюємо місцями 1 та 4 стовбці ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 1,1419 | 0,8273 | 0,6132 | 5,7988 | 0 | 5,9973 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,1725 | 2,8767 | 0,4262 | 0,8273 | 0 | 1,9990 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |
| 3,5781 | 0,1725 | 0,9556 | 1,1419 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 | 0,8349 | 0,3387 | 5,5369 | 0 | 0 | 3,7680 |

```
/// Ділимо рядок 2 на локальний максимум (5,7988) ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,1725 | 2,8767 | 0,4262 | 0,8273 | 0 | 1,9990 | 0,8349 | 0,3387 | 5,5369 | 0 | 0 | 3,7680 |
| 3,5781 | 0,1725 | 0,9556 | 1,1419 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |

```
/// Міняємо місцями 3 і 5 рядки ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 | 0,8349 | 0,3387 | 5,5369 | 0 | 0 | 3,7680 |
| 3,5781 | 0,1725 | 0,9556 | 1,1419 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |

```
/// Змінюємо місцями 3 та 3 стовбці ///
```

| | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 | 0,1875 | 0,3081 | 0,2282 | 0,1381 | 000001 | 0,5712 |
| 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 | 0,1969 | 0,1427 | 0,1057 | 000001 | 0 | 1,0342 |
| 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 | 0,8349 | 0,3387 | 5,5369 | 0 | 0 | 3,7680 |
| 3,5781 | 0,1725 | 0,9556 | 1,1419 | 0 | 5,5131 | 3,3533 | 0,0096 | 0,8349 | 0 | 0 | 4,3321 |
| 0,9556 | 0,4262 | 5,6017 | 0,6132 | 0 | 4,4032 | 0,0096 | 2,7587 | 0,3387 | 0 | 0 | 1,1434 |


```

/// Ділимо рядок 3 на локальний максимум (5,5369) ///
0,1875 0,3081 0,2282 0,1381 000001 0,5712 0,3081 0,1875 0,2282 0,1381 000001 0,5712
0,1969 0,1427 0,1057 000001 0 1,0342 0,1427 0,1969 0,1057 000001 0 1,0342
0,1508 0,0612 000001 0 0 0,6807 0,0612 0,1508 000001 0 0 0,6807
3,3533 0,0096 0,8349 0 0 4,3321 -0,041 3,2274 0 0 0 3,7638
0,0096 2,7587 0,3387 0 0 1,1434 2,7380 -0,041 0 0 0 0,9128

/// Робимо перетворення з рядком 4 ///
0,1875 0,3081 0,2282 0,1381 000001 0,5712 0,3081 0,1875 0,2282 0,1381 000001 0,5712
0,1969 0,1427 0,1057 000001 0 1,0342 0,1427 0,1969 0,1057 000001 0 1,0342
0,1508 0,0612 000001 0 0 0,6807 0,0612 0,1508 000001 0 0 0,6807
3,2274 -0,041 0 0 0 3,7638 -0,013 000001 0 0 0 1,1662
0,0096 2,7587 0,3387 0 0 1,1434 2,7380 -0,041 0 0 0 0,9128

/// Робимо перетворення з рядком 5 ///
0,1875 0,3081 0,2282 0,1381 000001 0,5712 0,3081 0,1875 0,2282 0,1381 000001 0,5712
0,1969 0,1427 0,1057 000001 0 1,0342 0,1427 0,1969 0,1057 000001 0 1,0342
0,1508 0,0612 000001 0 0 0,6807 0,0612 0,1508 000001 0 0 0,6807
3,2274 -0,041 0 0 0 3,7638 -0,013 000001 0 0 0 1,1662
-0,041 2,7380 0 0 0 0,9128 2,7375 0 0 0 0 0,9612

/// Міняємо місцями 4 і 4 рядки ///
0,1875 0,3081 0,2282 0,1381 000001 0,5712 0,3081 0,1875 0,2282 0,1381 000001 0,5712
0,1969 0,1427 0,1057 000001 0 1,0342 0,1427 0,1969 0,1057 000001 0 1,0342
0,1508 0,0612 000001 0 0 0,6807 0,0612 0,1508 000001 0 0 0,6807
3,2274 -0,041 0 0 0 3,7638 -0,013 000001 0 0 0 1,1662
-0,041 2,7380 0 0 0 0,9128 2,7375 0 0 0 0 0,9612

```

```

/// Змінюємо місцями 1 та 1 стовбці ///
0,3081 0,1875 0,2282 0,1381 000001 0,5712
0,1427 0,1969 0,1057 000001 0 1,0342
0,0612 0,1508 000001 0 0 0,6807
-0,013 000001 0 0 0 1,1662
2,7375 0 0 0 0 0,9612

/// Ділимо рядок 5 на локальний максимум (2,7375) ///
0,3081 0,1875 0,2282 0,1381 000001 0,5712
0,1427 0,1969 0,1057 000001 0 1,0342
0,0612 0,1508 000001 0 0 0,6807
-0,013 000001 0 0 0 1,1662
000001 0 0 0 0 0,3511

```

!!! Зворотний хід матриці Гауса !!!

Починаємо з останнього рядка

Беремо число B[5] = 0,3511

Тимчасовий результат: tempRes -= A[5][1] * result[2 - 1] = 0,3511

Тимчасовий результат: tempRes -= A[5][2] * result[4 - 1] = 0,3511

Тимчасовий результат: tempRes -= A[5][3] * result[5 - 1] = 0,3511

Тимчасовий результат: tempRes -= A[5][4] * result[1 - 1] = 0,3511

Тимчасовий результат: tempRes -= A[5][5] * result[3 - 1] = 0,3511

x2 = 0,3511

Беремо число B[4] = 1,1662

Тимчасовий результат: tempRes -= A[4][1] * result[2 - 1] = 1,1707

Тимчасовий результат: tempRes -= A[4][2] * result[4 - 1] = 1,1707

Тимчасовий результат: tempRes -= A[4][3] * result[5 - 1] = 1,1707

Тимчасовий результат: tempRes -= A[4][4] * result[1 - 1] = 1,1707

Тимчасовий результат: tempRes -= A[4][5] * result[3 - 1] = 1,1707

x4 = 1,1707

Беремо число B[3] = 0,6807

Тимчасовий результат: tempRes -= A[3][1] * result[2 - 1] = 0,6592

Тимчасовий результат: tempRes -= A[3][2] * result[4 - 1] = 0,4827

Тимчасовий результат: tempRes -= A[3][3] * result[5 - 1] = 0,4827

Тимчасовий результат: tempRes -= A[3][4] * result[1 - 1] = 0,4827

Тимчасовий результат: tempRes -= A[3][5] * result[3 - 1] = 0,4827

x5 = 0,4827

```

Беремо число B[2] = 1,0342
Тимчасовий результат: tempRes -= A[2][1] * result[2 - 1] = 0,9841
Тимчасовий результат: tempRes -= A[2][2] * result[4 - 1] = 0,7536
Тимчасовий результат: tempRes -= A[2][3] * result[5 - 1] = 0,7026
Тимчасовий результат: tempRes -= A[2][4] * result[1 - 1] = 0,7026
Тимчасовий результат: tempRes -= A[2][5] * result[3 - 1] = 0,7026
x1 = 0,7026

Беремо число B[1] = 0,5712
Тимчасовий результат: tempRes -= A[1][1] * result[2 - 1] = 0,4630
Тимчасовий результат: tempRes -= A[1][2] * result[4 - 1] = 0,2435
Тимчасовий результат: tempRes -= A[1][3] * result[5 - 1] = 0,1334
Тимчасовий результат: tempRes -= A[1][4] * result[1 - 1] = 0,0363
Тимчасовий результат: tempRes -= A[1][5] * result[3 - 1] = 0,0363
x3 = 0,0363

/// !!! МАСИВ РЕЗУЛЬТАТІВ !!! ///
x1 = 0,7026
x2 = 0,3511
x3 = 0,0363
x4 = 1,1707
x5 = 0,4827

```

Отримуємо масив результатів. Обчислюємо вектор нев'язки.

```

!!! Розв'язування вектора нев'язки !!!
Формула:  $r = B - Ax$ 

/// Розширена матриця ///
005,93  001,12  000,95  001,32  000,83          006,54
001,12  003,53  002,12  000,57  000,91          003,21
000,95  002,12  006,88  001,29  001,57          003,93
001,32  000,57  001,29  003,82  001,25          006,25
000,83  000,91  001,57  001,25  005,96          005,30

/// Вектор результатів ///
x1 = 0,7026
x2 = 0,3511
x3 = 0,0363
x4 = 1,1707
x5 = 0,4827

!!! Вектор нев'язки !!!
[-0,000  0,0000  0,0000  0,0000  -0,000  ]

```

Як бачимо, елементи вектору нев'язок близькі до 0 – а це означає, що результат близький до точного. Таке можливо через те, що в мові С# наявний тип даних decimal, який зберігає дробові числа не в двійковому форматі, а в десятичному. Також, на відміну від double, він займає не 8 байтів, а 16. Тобто, точність подвоєна.

Перевіримо отримані результати в MathCad:

ТР-12 Ковальов Олександр, варіант 22

$$var := 22$$

$$\alpha := 0.25 \quad |var - 25| = 0.75$$

$$\beta := 0.35 \quad (var - 21) = 0.35$$

$$A := \begin{bmatrix} 5.18 + \alpha & 1.12 & 0.95 & 1.32 & 0.83 \\ 1.12 & 4.28 - \alpha & 2.12 & 0.57 & 0.91 \\ 0.95 & 2.12 & 6.13 + \alpha & 1.29 & 1.57 \\ 1.32 & 0.57 & 1.29 & 4.57 - \alpha & 1.25 \\ 0.83 & 0.91 & 1.57 & 1.25 & 5.21 + \alpha \end{bmatrix} = \begin{bmatrix} 5.93 & 1.12 & 0.95 & 1.32 & 0.83 \\ 1.12 & 3.53 & 2.12 & 0.57 & 0.91 \\ 0.95 & 2.12 & 6.88 & 1.29 & 1.57 \\ 1.32 & 0.57 & 1.29 & 3.82 & 1.25 \\ 0.83 & 0.91 & 1.57 & 1.25 & 5.96 \end{bmatrix}$$

$$B := \begin{bmatrix} 6.19 + \beta \\ 3.21 \\ 4.28 - \beta \\ 6.25 \\ 4.95 + \beta \end{bmatrix} = \begin{bmatrix} 6.54 \\ 3.21 \\ 3.93 \\ 6.25 \\ 5.3 \end{bmatrix} \quad X_m := A^{-1} \cdot B = \begin{bmatrix} 0.7026 \\ 0.3511 \\ 0.0363 \\ 1.1707 \\ 0.4827 \end{bmatrix} \quad X := \begin{bmatrix} 0.7026 \\ 0.3511 \\ 0.0363 \\ 1.1707 \\ 0.4827 \end{bmatrix} \quad \begin{matrix} dim := 5 \\ // \text{Розмірність} \end{matrix}$$

$$r := B - A \cdot X_m = \begin{bmatrix} 8.882 \cdot 10^{-16} \\ 0 \\ 4.441 \cdot 10^{-16} \\ 8.882 \cdot 10^{-16} \\ 8.882 \cdot 10^{-16} \end{bmatrix} \quad \delta := \sqrt[2]{\left(\frac{1}{dim}\right) \cdot \sum_{k=0}^{dim-1} (X_k - X_{m_k})^2} = 3.58 \cdot 10^{-5}$$

Як бачимо, вектор X повністю співпадає (відображаються 6 значимих чисел). Всі елементи вектору нев'язки також дуже близькі до нуля. Була обчислена похибка за формулою, і можна сказати, що результат відмінний.

Висновок: за допомогою мови програмування С# був написаний метод Гауса. Програма була протестована на даних з індивідуального завдання. Для перевірки результатів використовувалося програмне забезпечення MathCad. Були отримані хороші результати.

Додатки.

Код програми (Replit): <https://replit.com/join/sfvwuwfoua-kovalevalex>

Код програми:

Program.cs

```
namespace Lab1;

class Program
{
    public static void Main(string[] args)
    {
        const int var = 22;

        var a = 0.25m * Math.Abs(var - 25);
        var b = 0.35m * (var - 21);

        decimal[][] A =
        {
            new decimal[] { 5.18m + a, 1.12m, 0.95m, 1.32m, 0.83m },
            new decimal[] { 1.12m, 4.28m - a, 2.12m, 0.57m, 0.91m },
            new decimal[] { 0.95m, 2.12m, 6.13m + a, 1.29m, 1.57m },
            new decimal[] { 1.32m, 0.57m, 1.29m, 4.57m - a, 1.25m },
            new decimal[] { 0.83m, 0.91m, 1.57m, 1.25m, 5.21m +
a }

        };

        decimal[] B = { 6.19m + b, 3.21m, 4.28m - b, 6.25m, 4.95m + b };

        decimal[][] ACopy = Utils.CopyMatrix(A);
        decimal[] BCopy = Utils.CopyArray(B);

        int[] xpos = new int[B.Length];

        ForwardElimination.Start(A, B, xpos);
        decimal[] result = BackSubstitution.Start(A, B, xpos);

        ResidualVector.Start(ACopy, BCopy, result);
    }
}

/*
Matrix from the video:
decimal[][] A =
{
    new decimal[] { 1, 1, -1 },
    new decimal[] { 2, 1, 2 },
    new decimal[] { 1, -3, 1 }
};

decimal[] B = { 0, 10, -2 };
*/
```

ForwardElimination.cs

```
namespace Lab1;

public class ForwardElimination
{
    public static void Start(decimal[][] A, decimal[] B, int[] xpos)
    {
        // filling x array (1 2 3 4 5 ....)
        for (int i = 0; i < xpos.Length; i++)
        {
```

```

        xpos[i] = i + 1;
    }

    Printer.PrintResultMatrix(A, B, "Початкова матриця, початок прямого ходу
методу Гауса");

    // i == limiter
    for (int i = 0; i < A.Length; i++)
    {
        int rowWMax = Utils.RowWMax(A, i);
        Utils.Swap(i, rowWMax, A);
// swap row in A
        Utils.Swap(i, rowWMax, B);
// swap row in B

        Printer.PrintResultMatrix(A, B, $"Міняємо місцями {i + 1} і {rowWMax
+ 1} рядки");

        int initialMaxId = Utils.IndexOfElement(Utils.Max(A[i], i), A[i],
i);
        Utils.Swap(A[i].Length - 1 - i, initialMaxId, xpos);
// swap x elems
        for (int j = 0; j < A.Length; j++)
        {
            Utils.Swap(A[j].Length - 1 - i, initialMaxId, A[j]);
// swap elems of arr
        }

        Printer.PrintResultMatrix(A, B, $"Змінюємо місцями {initialMaxId +
1} та {A[i].Length - i} стовбці");

        decimal rowMax = A[i][A[i].Length - 1 - i];
        for (int j = 0; j < A[i].Length - i; j++)
// dividing all elements on max
        {
            A[i][j] /= rowMax;
        }
        B[i] /= rowMax;

        Printer.PrintResultMatrix(A, B,
        $"Ділимо рядок {i + 1} на локальний максимум
({Printer.SignificantFigures(rowMax)})");

        for (int row = i + 1; row < A.Length; row++)
// other rows
        {
            var localMax = A[row][A[i].Length - 1 - i];

            for (int elem = 0; elem < A[row].Length - i; elem++)
// elems of row
            {
                A[row][elem] -= A[i][elem] * localMax;
            }

            B[row] -= B[i] * localMax;

            Printer.PrintResultMatrix(A, B, $"Робимо перетворення з рядком
{row + 1}");
        }
    }
}
}

```

BackSubstitution.cs

```
namespace Lab1;

public class BackSubstitution
{
    public static decimal[] Start(decimal[][] A, decimal[] B, int[] xpos)
    {
        Console.WriteLine("!!! Зворотний хід матриці Гауса !!!\n");
        var result = new decimal[A[0].Length];

        Console.WriteLine("Починаємо з останнього рядка");
        for (int i = A.Length - 1, counter = 0; i >= 0; i--, counter++)
        {
            decimal tempRes = B[i];
            Console.WriteLine($"Беремо число B[{i+1}] = {Printer.SignificantFigures(tempRes)}");
            for (int j = 0; j < A[i].Length; j++)
            {
                tempRes -= A[i][j] * result[xpos[j] - 1];
                Console.WriteLine($"Тимчасовий результат: tempRes -= A[{i+1}][{j+1}] * result[{xpos[j]} - 1] = " +
                    $"{Printer.SignificantFigures(tempRes)}");
            }

            result[xpos[counter] - 1] = tempRes;
            Console.WriteLine($"x{xpos[counter]} = {Printer.SignificantFigures(tempRes)}\n");
        }

        Printer.PrintResultArray(result, "!!! МАСИВ РЕЗУЛЬТАТІВ !!!");

        return result;
    }
}
```

ResidualVector.cs

```
namespace Lab1;

public class ResidualVector
{
    public static void Start(decimal[][] A, decimal[] B, decimal[] x)
    {
        Console.WriteLine("!!! Розв'язування вектора нев'язки !!!");
        Console.WriteLine("Формула:  $r = B - Ax$ \n");

        Printer.PrintResultMatrix(A, B, "Розширена матриця");
        Printer.PrintResultArray(x, "Вектор результатів");

        decimal[] r = Subtract(B, Mult(A, x));
        Console.WriteLine("!!! Вектор нев'язки !!!");
        Console.WriteLine("[");
        foreach (var elem in r)
        {
            Console.Write(Printer.SignificantFigures(elem) + "\t");
        }
        Console.WriteLine("]");

        // multiplication matrix * vector
        public static decimal[] Mult(decimal[][] a, decimal[] b)
        {
            if (a[0].Length != b.Length) throw new ArithmeticException("a[0].Length != b.Length");
        }
    }
}
```

```

        var result = new decimal[a.Length];

        for (int i = 0; i < a.Length; i++) {
            for (int j = 0; j < b.Length; j++) {
                decimal expr = 0;
                for (int k = 0; k < a[0].Length; k++) {
                    expr += a[i][k] * b[k];
                }
                result[i] = expr;
            }
        }
        return result;
    }

    // subtracting vectors
    public static decimal[] Subtract(decimal[] a, decimal[] b)
    {
        if (a.Length != b.Length && a.Length != b.Length) throw new
ArithmeticException("a[0].Length != b.Length");

```

```

        var result = new decimal[a.Length];

        for (int i = 0; i < a.Length; i++) {
            result[i] = a[i] - b[i];
        }
        return result;
    }
}

```

Utils.cs

```

using System.Text;
namespace Lab1;

public class Utils
{
    public static decimal Max(decimal[] arr, int limiter)
    {
        var max = arr[0];
        for (int i = 0; i < arr.Length - limiter; i++)
        {
            if (arr[i] > max)
            {
                max = arr[i];
            }
        }

        return max;
    }

    public static int IndexOfElement(decimal elem, decimal[] arr, int limiter)
    {
        for (int i = 0; i < arr.Length - limiter; i++)
        {
            if (arr[i].Equals(elem)) return i;
        }

        return -1;
    }

    public static int RowWMax(decimal[][] arr, int limiter)
    {
        var max = arr[limiter][0];
        var row = limiter;
        for (int i = limiter; i < arr.Length; i++)
        {

```

```

        var rowMax = Max(arr[i], limiter);
        if (rowMax > max)
        {
            max = rowMax;
            row = i;
        }
    }

    return row;
}

public static void Swap<T>(int i, int j, T[] arr)
{
    (arr[i], arr[j]) = (arr[j], arr[i]);
}

public static T[][] CopyMatrix<T>(T[][] matrix)
{
    var copy = new T[matrix.Length][];
    for (int i = 0; i < matrix.Length; i++)
    {
        copy[i] = new T[matrix[0].Length];
    }

    for (int i = 0; i < copy.Length; i++)
    {
        for (int j = 0; j < copy.Length; j++)
        {
            copy[i][j] = matrix[i][j];
        }
    }

    return copy;
}

public static T[] CopyArray<T>(T[] matrix)
{
    var copy = new T[matrix.Length];

    for (int i = 0; i < copy.Length; i++)
    {
        copy[i] = matrix[i];
    }

    return copy;
}
}

```

Printer.cs

```

using System.Text;

namespace Lab1;

public class Printer
{
    static Printer()
    {
        Console.OutputEncoding = Encoding.UTF8;
    }

    public static void PrintResultArray(decimal[] arr, params string[] messages)
    {
        if (messages.Length == 1) Console.WriteLine("/// " + messages[0] + "
        for (int i = 0; i < arr.Length; i++)
        ///");
    }
}

```



```

        {
            Console.Write($"x{i+1} = " + SignificantFigures(arr[i]) + "\n");
        }
        Console.WriteLine();
    }

    public static void PrintResultMatrix(decimal[][] A, decimal[] B, params
string[] messages)
    {
        if (messages.Length == 1) Console.WriteLine("/// " + messages[0] + "
///");
        for (int i = 0; i < A.Length; i++)
        {
            for (int j = 0; j < A[0].Length; j++)
            {
                Console.Write(SignificantFigures(A[i][j]) + "\t");
            }
            Console.Write("\t" + SignificantFigures(B[i]));
            Console.WriteLine();
        }
        Console.WriteLine();
    }

    public static string SignificantFigures(decimal number)
    {
        if (number == 0) return "0";

        const int significant = 6;

        var numberString = String.Format("{0}", number);
        var charArray = numberString.ToCharArray();
        var sb = new StringBuilder();

        int integer = 0;
        int fractional = 0;
        int comaPos = 0;

        for (int i = 0; i < charArray.Length; i++)
        {
            if (charArray[i] == ',') comaPos = i;

            if (charArray[i] == '-' || comaPos <= 0)
            {
                integer++;
            }
            else
            {
                fractional++;
            }
        }

        int possibleZeros = significant - fractional - integer;

        for (int i = 0, current = 0; i < significant && i < charArray.Length;
i++)
        {
            if (i == 0 && charArray[i] == '-')
            {
                sb.Append('-');
                current++;
                continue;
            }

            for (; possibleZeros >= 1; possibleZeros--)

```

```

        {
            sb.Append('0');
            current++;
        }

        if (current != significant - 1)
        {
            sb.Append(charArray[i]);
            current++;
        }
        else if (i + 1 < charArray.Length)
        {
            if (int.Parse(charArray[i + 1].ToString()) > 5)
            {
                sb.Append((int.Parse(charArray[i].ToString()) + 1) % 10);
            }
            else sb.Append(charArray[i]);
        }
        else
        {
            sb.Append(charArray[i]);
        }
    }

    return sb.ToString();
}

```