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 №2

з дисципліни «Чисельні методи»

Тема «Розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) ітераційними методами. Метод простої ітерації. Метод Зейделя»

Варіант №22

Студента 2-го курсу НН ІАТЕ гр. ТР-12

Ковальова Олександра

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

КИЇВ 2022

Мета роботи. Навчитися розв’язувати системи лінійних алгебраїчних рівнянь ітераційними методами. Написати програму для обчислення СЛАР для розв’язку матриці за варіантом. Перевірити отримані результати з результатами з онлайн калькулятору/Mathcad. Знайти середньоквадратичну похибку обчислень.

Теоретична частина.

Ітераційними методами є такі, що навіть у припущенні, що обчислення ведуться без округлень, дозволяють отримати розв’язок системи лише із заданою точністю. До таких методів відносяться метод простої ітерації (метод Якобі) та метод Зейделя.

Будемо розглядати системи вигляду:

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

Де $A(n \times n)$ – матриця системи, b – вектор правої частини, x – вектор розв’язку

<u>*Метод простої ітерації*</u>

Систему $Ax = b$ приводять до вигляду

$$x = Cx + d, \qquad (2)$$

Де C – деяка матриця, для якої виконується

$$\alpha = \max_i \sum_{j=1}^{n} |c_{ij}| < 1 \text{ або } \alpha = \max_j \sum_{i=1}^{n} |c_{ij}| < 1 \text{ або } \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij})^2 < 1, \qquad (3)$$

d – вектор-стовпець.

Умова (3) буде виконана, якщо матриця A є матрицею з діагональною перевагою, для якої $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ або $|a_{jj}| > \sum_{j \neq i} |a_{ij}|$

Розглянемо спосіб зведення (1) до (2). Запишемо (1) у розгорнутій формі:

$$-\sum_{j=1}^{n} a_{ij}x_j + b_i = 0, \qquad i = \overline{1,n}, \qquad (4)$$

Якщо $a_{ij} \neq 0$ для всіх i, то можна (4) зобразити у вигляді

$$x_i = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j + \frac{b_i}{a_{ii}}, \qquad i = \overline{1,n}, \qquad (5)$$

Звідси отримуємо значення елементів матриці C та вектору d:

$$c_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}}, & i \neq j \\ 0, & i = j \end{cases}, \qquad d_i = \frac{b_i}{a_{ii}}, \qquad i = \overline{1,n}$$

Запишемо розв’язок у матричному вигляді. Нехай матрицю A задано у вигляді:

$$A = A_1 + D + A_2,$$

де A_1 – нижня трикутна матриця з нульовою головною діагоналлю; D – діагональна матриця з aii на головній діагоналі; A_2 – верхня трикутна матриця з нульовою головною діагоналлю. За припущенням $a_{ij} \neq 0$ для всіх i, існує D^{-1}. Тоді зображенню у формі (5) відповідає:

$$x = -D^{-1}A_1x - D^{-1}A_2x - D^{-1}b$$

Або

$$x = -D^{-1}(A_1 + A_2)x - D^{-1}b.$$

Якщо матриця A не забезпечує виконання (3), тобто не є матрицею з діагональною перевагою, її приводять до такої за допомогою еквівалентних перетворень.

Виходячи з довільного вектора $x^{(0)}$ (можна взяти вектор b, або вектор b, поділений на діагональ матриці A) будують ітераційний процес:

$$x^{(k+1)} := Cx^{(k)} + d$$

Або

$$x^{(k+1)} == -D^{-1}(A_1 + A_2)x^{(k)} - D^{-1}b$$

Критерій закінчення ітераційного процесу:

$$\max_j |x_j^{k+1} - x_j^k| < \varepsilon.$$

<u>*Метод Зейделя*</u>

Цей метод – модифікація методу простої ітерації. В цьому методі вже знайдені компоненти беруть у правій частині співвідношення з (n+1)-го наближення, а іншні – з n-го наближення:

$$x_i^{(k+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}, \qquad i = \overline{1,n}.$$

Або у матричному вигляді:

$$x^{(k+1)} = -D^{-1}A_1x^{(k+1)} - D^{-1}A_2x^{(k)} + D^{-1}b.$$

Умови застосування методу Зейделя, критерій закінчення ітерацій такі самі, як для методу простої ітерації.

Завдання. Якщо матриця не є матрицею із діагональною перевагою, привести систему до еквівалентної, у якій є діагональна перевага. Можна, наприклад, провести одну ітерацію метода Гауса, скомбінувавши рядки з метою отримати нульовий недіагональний елемент у стовпчику. Розробити програму, що реалізує розв'язання за ітераційним методом, який відповідає заданому варіанту. Обчислення проводити з кількістю значущих цифр $m = 6$. Для кожної ітерації розраховувати нев'язку $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax}$, де x - отриманий розв'язок.

Розв'язати задану систему рівнянь за допомогою програмного забезпечення Mathcad. Навести результат перевірки: вектор нев'язки $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax_m}$, де x_m - отриманий у Mathcad розв'язок.

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

Хід роботи

Індивідуальне завдання:

Розв'язати матрицю методом простих ітерацій.

Матриця:

$$A = \begin{pmatrix} 8{,}30 & 2{,}62+\alpha & 4{,}10 & 1{,}90 \\ 3{,}92 & 8{,}45 & 8.78-\alpha & 2{,}46 \\ 3{,}77 & 7{,}21+\alpha & 8{,}04 & 2{,}28 \\ 2{,}21 & 3{,}65-\alpha & 1{,}69 & 6{,}99 \end{pmatrix}, \qquad \alpha = 0.2k, \qquad k = \text{№}_{\text{варіанту}} - 22$$

$$b = \begin{pmatrix} -10{,}65 + \beta \\ 12{,}21 \\ 15{,}45 - \beta \\ -8{,}35 \end{pmatrix}, \qquad \beta = 0.2k, \qquad k = №_{\text{варіанту}} - 22$$

α	β	k
$0.2k = 0.2 \cdot 0 = 0.0$	$0.2k = 0.2 \cdot 0 = 0.0$	$22 - 22 = 0$

Матриця відповідно до варіанту:

$$\kappa := 22 - 22 = 0$$
$$\alpha := 0.2 \cdot \kappa = 0$$
$$\beta := 0.2 \cdot \kappa = 0$$

$$A := \begin{bmatrix} 8.30 & 2.62 & 4.10 & 1.90 \\ 3.92 & 8.45 & 8.78 & 2.46 \\ 3.77 & 7.21 & 8.04 & 2.28 \\ 2.21 & 3.65 & 1.69 & 6.99 \end{bmatrix} \qquad b := \begin{bmatrix} -10.65 \\ 12.21 \\ 15.45 \\ -8.35 \end{bmatrix}$$

Перевіримо, чи наша матриця є матрицею з діагональною перевагою:

$$A = \begin{pmatrix} 8{,}30 & 2{,}62 & 4{,}10 & 1{,}90 \\ 3{,}92 & 8{,}45 & 8.78 & 2{,}46 \\ 3{,}77 & 7{,}21 & 8{,}04 & 2{,}28 \\ 2{,}21 & 3{,}65 & 1{,}69 & 6{,}99 \end{pmatrix},$$

$$|8{,}30| >?\ 2{,}62 + 4{,}10 + 1{,}90 \geq?\ |8{,}30| > 8{,}62 - \text{Ні}$$
$$|8{,}45| >?\ 3{,}92 + 8{,}78 + 2{,}46\ \geq?\ |8{,}45| > 15{,}16 - \text{Ні}$$
$$|8{,}04| >?\ 2{,}28 + 3{,}77 + 7{,}21\ \geq?\ |8{,}04| > 13{,}26 - \text{Ні}$$
$$|6{,}99| >?\ 2{,}21 + 1{,}69 + 3{,}65\ \geq?\ |6{,}99| > 7{,}55 - \text{Ні}$$

Бачимо, що дана матриця не є матрицею з діагональною перевагою. Застосуємо елементарні перетворення для приведення матриці до необхідного нам виду

1) 2-й рядок – 3-й ряд

$$A = \begin{pmatrix} 8{,}30 & 2{,}62 & 4{,}10 & 1{,}90 \\ 0{,}15 & 1{,}24 & 0{,}74 & 0{,}18 \\ 3{,}77 & 7{,}21 & 8{,}04 & 2{,}28 \\ 2{,}21 & 3{,}65 & 1{,}69 & 6{,}99 \end{pmatrix}, b = \begin{pmatrix} -10{,}65 \\ -3{,}24 \\ 15{,}45 \\ -8{,}35 \end{pmatrix}$$

2) 1-й ряд – 2-й рядок*2, 4-й рядок – 2-й рядок*2

$$A = \begin{pmatrix} 8{,}00 & 0{,}14 & 2{,}62 & 1{,}54 \\ 0{,}15 & 1{,}24 & 0{,}74 & 0{,}18 \\ 3{,}77 & 7{,}21 & 8{,}04 & 2{,}28 \\ 1{,}91 & 1{,}17 & 0{,}25 & 6{,}63 \end{pmatrix}, b = \begin{pmatrix} -4{,}17 \\ -3{,}24 \\ 15{,}45 \\ -1{,}87 \end{pmatrix}$$

2) 3-й рядок – 2-й рядок*7

$$A = \begin{pmatrix} 8{,}00 & 0{,}14 & 2{,}62 & 1{,}54 \\ 0{,}15 & 1{,}24 & 0{,}74 & 0{,}18 \\ 2{,}72 & -1{,}47 & 2{,}86 & 1{,}02 \\ 1{,}91 & 1{,}17 & 0{,}25 & 6{,}63 \end{pmatrix}, b = \begin{pmatrix} -4{,}17 \\ -3{,}24 \\ 38{,}13 \\ -1{,}87 \end{pmatrix}$$

Запуск програми.

Програма для обчислень була написана мовою програмування C# у оточенні розробки JetBrains Rider. Версія мови – C# 11, платформа – .NET 7.

Спочатку виводимо матрицю з діагональною перевагою та стовпчик відповідей:

```
--- Kovalov Alex, TP-12 ---
        -- Lab 2 --

A:
x1 = 008,00     x2 = 000,14     x3 = 002,62     x4 = 001,54
x1 = 000,15     x2 = 001,24     x3 = 000,74     x4 = 000,18
x1 = 002,72     x2 = -01,47     x3 = 002,86     x4 = 001,02
x1 = 001,91     x2 = 001,17     x3 = 000,25     x4 = 006,63

B:
-04,17  -03,24  038,13  -01,87
```

Після цього запускаємо метод простих ітерацій. Будуть виводитися лише вектори нев’язок.

```
Calculations by method of simple iterations started!

Residual vectors:
Iteration 1:    -34,13  -9,737  -2,135  0,7196
Iteration 2:    2,8884  1,1729  -0,049  17,522
Iteration 3:    -4,157  -0,517  -2,287  -1,792
Iteration 4:    2,5700  0,7184  1,0761  1,6804
```

Коли вектор нев’язок переходить межу точності, закінчуємо підрахунок та виводимо результат (6 значимих чисел):

```
Iteration 22:   0,0000  0,0000  0,0000  0,0000
Iteration 23:   -0,000  -0,000  -0,000  -0,000
Iteration 24:   0,0000  0,0000  0,0000  0,0000

X[1] = -4,764
X[2] = -9,596
X[3] = 12,101
X[4] = 2,3276

Iterations done: 24
Accuracy: 0,00001
```

Результати від сервісу-калькулятора:

Розв'язок СЛАР методом простої ітерації

13	-4.764	-9.595	12.102	2.329	-0.00262	-0.00408	0.00253	0.00291
14	-4.765	-9.597	12.101	2.327	0.00146	0.00225	-0.00143	-0.00157
15	-4.764	-9.596	12.101	2.328	-0.000812	-0.00126	0.000792	0.000871
16	-4.765	-9.596	12.101	2.327	0.000449	0.000697	-0.000434	-0.000486

Середньо-квадратичну похибку порахуємо за допомогою Mathcad.
A – матриця з діагональною перевагою.
B – стовпчик відповідей.
X_m – корені, здобуті за допомогою Mathcad.
X – корені, здобуті за допомогою власноруч написаної програми.
r – вектор нев’язки.

ТР-12 Ковальов Олександр, варіант 22

$$A := \begin{bmatrix} 8 & 0.14 & 2.62 & 1.54 \\ 0.15 & 1.24 & 0.74 & 0.18 \\ 2.72 & -1.47 & 2.86 & 1.02 \\ 1.91 & 1.17 & 0.25 & 6.63 \end{bmatrix} \quad B := \begin{bmatrix} -4.17 \\ -3.24 \\ 38.13 \\ -1.87 \end{bmatrix} \quad Xm := A^{-1} \cdot B = \begin{bmatrix} -4.7645 \\ -9.596 \\ 12.1011 \\ 2.3276 \end{bmatrix} \quad X := \begin{bmatrix} -4.764 \\ -9.596 \\ 12.101 \\ 2.3276 \end{bmatrix}$$

$$r := B - A \cdot Xm = \begin{bmatrix} 3.553 \cdot 10^{-15} \\ 2.22 \cdot 10^{-15} \\ 7.105 \cdot 10^{-15} \\ 6.217 \cdot 10^{-15} \end{bmatrix} \qquad dim := 4 \qquad \delta := \sqrt[2]{\left(\frac{1}{dim}\right) \cdot \sum_{k=0}^{dim-1} \left(X_k - Xm_k\right)^2} = 247.68 \cdot 10^{-6}$$

Отримуємо середньо-квадратичну похибку, яка дорівнює 0.2%.

Висновок. Під час виконання лабораторної роботи була розроблена програма, яка вирішує СЛАР методом ітерацій. Також, було ознайомлено з суттю цього методу.

Додатки.

Код програми (Replit): https://replit.com/join/mqnqmdmuth-kovalevalex

Код програми:

Program.cs

```
namespace Lab2;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("--- Kovalov Alex, TP-12 ---");
        Console.WriteLine("        -- Lab 2 --      \n");

        decimal[][] A = {
            new[] {8.00m, 0.14m, 2.62m, 1.54m},
            new[] {0.15m, 1.24m, 0.74m, 0.18m},
            new[] {2.72m, -1.47m, 2.86m, 1.02m},
            new[] {1.91m, 1.17m, 0.25m, 6.63m}
        };

        decimal[] B = { -4.17m, -3.24m, 38.13m, -1.87m };

        Console.WriteLine("A:");
        Printer.Matrix(A);

        Console.WriteLine("B:");
        Printer.Vector(B);

        Console.WriteLine();

        Calculations.Iterations(A, B);
    }
}
```

Calculations.cs

```
namespace Lab2;

public class Calculations
{
    public static void Iterations(decimal[][] A, decimal[] B, decimal eps =
0.00001m, int Length = 4)
    {
        Console.WriteLine("Calculations by method of simple iterations
started!\n");

        Console.WriteLine("Residual vectors:");

        var X0 = new decimal[Length];          // Start values
        var X = new decimal[Length];           // Result
        var C = new decimal[Length];           // Multiplying A*x

        int count = 0;                         // Iteration counter

        var temp = new decimal[Length];        // For accuracy
        decimal maxTemp;

        var Residual = new decimal[Length];    // Residual vector

        do
        {
```

```
                    count++;

                    // Method
                    X[0] = (B[0] - A[0][1] * X0[1] - A[0][2] * X0[2] - A[0][3] * X0[3])
/ A[0][0];
                    X[1] = (B[1] - A[1][0] * X0[0] - A[1][2] * X0[2] - A[1][3] * X0[3])
/ A[1][1];
                    X[2] = (B[2] - A[2][0] * X0[0] - A[2][1] * X0[1] - A[2][3] * X0[3])
/ A[2][2];
                    X[3] = (B[3] - A[3][0] * X0[0] - A[3][1] * X0[1] - A[3][2] * X0[2])
/ A[3][3];

                    // Subtraction of X
                    for (int i = 0; i < temp.Length; i++)
                    {
                        temp[i] = Math.Abs(X[i] - X0[i]);
                    }

                    // Max from subtraction of results
                    maxTemp = temp.Max();

                    // Init new values
                    for (int i = 0; i < Length; i++)
                    {
                        X0[i] = X[i];
                    }

                    // Residual vector solving
                    // C = A * x
                    decimal tempValue = 0;
                    for (int i = 0; i < Length; i++)
                    {
                        for (int j = 0; j < Length; j++)
                        {
                            tempValue += A[i][j] * X[j];
                        }

                        C[i] = tempValue;
                        tempValue = 0;
                    }

                    // R = B - C
                    for (int i = 0; i < Length; i++)
                    {
                        Residual[i] = B[i] - C[i];
                    }

                    Console.Write($"Iteration {count}: \t");
                    Printer.Vector(Residual);
                } while (maxTemp > eps);

                Console.WriteLine();
                for (int i = 0; i < X.Length; i++)
                {
                    Console.WriteLine($"X[{i + 1}] =
{Printer.SignificantFigures(X[i])}");
                }
                Console.WriteLine();


                Console.WriteLine($"Iterations done: {count}");
                Console.WriteLine($"Accuracy: {eps}");
            }
        }
```

Printer.cs

```
using System.Text;

namespace Lab2;

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

    public static void Matrix(decimal[][] arr, params string[] messages)
    {
        if (messages.Length == 1) Console.WriteLine($"/// {messages[0]} ///");
        foreach (var row in arr)
        {
            for (int i = 0; i < arr.Length; i++)
            {
                Console.Write($"x{i + 1} = {SignificantFigures(row[i])}\t");
            }
            Console.WriteLine();
        }
        Console.WriteLine();
    }

    public static void Vector(decimal[] arr, params string[] messages)
    {
        if (messages.Length == 1) Console.WriteLine($"/// {messages[0]} ///");

        foreach (var t in arr)
        {
            Console.Write($"{SignificantFigures(t)}  ");
        }
        Console.WriteLine();
    }

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

        const int significant = 6;

        var numberString = $"{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();
        }
    }
```