Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ

к лабораторной работе

на тему

Численное решение систем линейных уравнений методом простых итераций и методом Зейделя

Выполнил: студент группы 153501
Тимофеев Кирилл Андреевич

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Вариант 8

Цель выполнения задания:

- изучить иттъерационные методы решения СЛАУ (метод простых итераций, метод Зейделя);
- составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
- составить программу решения СЛАУ по разработанному алгоритму;
- Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоёмкость решения методом простых итераций и методом Зейделя.

Краткие теоретические сведения:

Итерационные методы основаны на построении сходящейся к точном решению $\boldsymbol{x}$ рекуррентной последовательности.

Для решения СЛАУ **методом простых итераций** преобразуем систему от первоначальной формы $\boldsymbol{Ax} = \boldsymbol{b}$ или

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1,$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2,$$

$$\ldots..$$

(2.1)

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$$

к виду

$$\boldsymbol{x} = \boldsymbol{Bx} + \boldsymbol{c}. \qquad (2.2)$$

Задав произвольным образом столбец начальных приближений $\boldsymbol{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)^T$, подставим их в правые части системы (2.2) и вычислим новые приближения $\boldsymbol{x}^1 = (x_1^1, x_2^1, \ldots, x_n^1)^T$, которые опять подставим в систему (2.2) и т.д. Таким образом, организуется итерационный процесс

Метод Зейделя. Метод Зейделя является модификацией метода простых итераций. Суть его состоит в том, что при вычислении следующего x_i^k : 2 $\le 1 \le n$ в формуле $\boldsymbol{x^k = Bx^{k-1} + c}$, k = 1,2, ... используются вместо $x_1^{k-1},\dots,x_{i-1}^{k-1}$ уже вычисленные ранее $x_1^k,\dots,x_{i-1}^k$, т.е.

$$x_i^k = \sum_{j=1}^{i-1} g_{ij} x_j^k + \sum_{j=i+1}^{n} g_{ij} x_j^{k-1} + c_i \quad (2.3)$$

Такое усовершенствование позволяет ускорить сходимость итераций почти в два раза. Кроме того, данный метод может быть реализован на ЭВМ без привлечения дополнительного массива, т.к. полученное новое x_i^k сразу засылается на место старого.

Схема алгоритма аналогична схеме метода простых итераций.

Програмная реализация:

```
using System;
using System.IO;
namespace MChA2{
    class Program{
        static string projectLocation = "D:\\Mocha\\MChA2\\";
        static int rows = 5, columns = 5;
        static decimal epsilon = 0.0000005M;
        static void Main(string[] args){
            int k = 8;
            Matrix C = new Matrix(rows, columns);
            ReadC(C, 0);
            Matrix D = new Matrix(rows, columns);
            ReadD(D, 0);
            Matrix B = new Matrix(rows, 1);
            ReadB(B, 0);

            Matrix A = k * C + D;
            Console.WriteLine("Матрица A\n");
```

```
        Console.WriteLine(A.ToString() + "\n\n");
        Console.WriteLine("\nМатрица B\n");
        Console.WriteLine(B.ToString() + "\n\n");
        Console.WriteLine("\n\nПрямая иттерация\n");
        straightItteration(A, B);
        Console.WriteLine("\n\nМетод Зейделя");
        Zeidel(A, B);
    }
    static void ReadD(Matrix D, int mode){
        string[] DPath = new string[2] { "DDefault.txt", "D.txt" };
        StreamReader sr = new StreamReader(projectLocation + DPath[mode]);
        for (int i = 0; i < D.Rows; ++i){
            string line = sr.ReadLine();
            for (int j = 0; j < D.Columns; ++j){
                int f = line.IndexOf(' ');
                //Console.WriteLine("f = " + f);
                if (f != -1) D[i, j] = Convert.ToDecimal(line.Substring(0, f));
                else D[i, j] = Convert.ToDecimal(line);
                line = line.Substring(f + 1);
            }
        }
    }
    static void ReadB(Matrix B, int mode){
        string[] BPath = new string[2] { "BDefault.txt", "B.txt" };
        StreamReader sr = new StreamReader(projectLocation + BPath[mode]);
        string line = sr.ReadLine();
        for (int i = 0; i < B.Rows; ++i){
            int f = line.IndexOf(' ');
            if (f != -1) B[i, 0] = Convert.ToDecimal(line.Substring(0, f));
            else B[i, 0] = Convert.ToDecimal(line);
            line = line.Substring(f + 1);
        }
    }
    static void ReadC(Matrix C, int mode){
        string[] CPath = new string[2] { "DDefault.txt", "D.txt" };
        StreamReader sr = new StreamReader(projectLocation + CPath[mode]);
```

```
    for (int i = 0; i < rows; ++i){
        string line = sr.ReadLine();

        for (int j = 0; j < columns; ++j){
            int f = line.IndexOf(' ');
            if (f != -1) C[i, j] = Convert.ToDecimal(line.Substring(0, f));
            else C[i, j] = Convert.ToDecimal(line);
            line = line.Substring(f + 1);
        }
    }
}
static void straightItteration(Matrix _A, Matrix _B){
    Matrix A = -_A;
    Matrix B = new Matrix(_B);

    for (int i = 0; i < _A.Rows; ++i){
        decimal a = -A[i, i];
        A[i, i] = B[i, 0];
        for (int j = 0; j < _A.Columns; ++j){
            A[i, j] /= a;
        }
    }
    Matrix xk = new Matrix(A.Rows, 1);
    Matrix xk_1 = new Matrix(A.Rows, 1);
    for (int i = 0; i < B.Rows; ++i) xk[i,0] = A[i,i];
    int count = 0;
    while (Math.Abs((xk - xk_1).norm()) >= epsilon){

        for (int i = 0; i < A.Columns; ++i) xk_1[i, 0] = xk[i, 0];

        for (int i = 0; i < A.Rows; ++i){
            xk[i, 0] = 0;
            for(int j = 0; j < A.Columns; ++j){
                if (i != j) xk[i, 0] += A[i, j] * xk_1[j, 0];
                else xk[i, 0] += A[i, j];
            }
```

```
                }
                ++count;
            }
            for (int i = 0; i < xk.Rows; ++i)
              Console.WriteLine("x" + (i + 1) + " = " + xk[i, 0] + "\n");
            Console.WriteLine("Подстановка:\n");
            for(int i = 0; i < A.Rows; ++i){
                decimal sum = 0;
                for(int j = 0; j < _A.Columns; ++j){
                    sum += _A[i, j] * xk[j, 0];
                }
                Console.WriteLine((i + 1 ) + ". " + sum + " = " + B[i, 0] + " => невязка = " + (sum
- B[i, 0]) + "\n");
            }
        }
        static void Zeidel(Matrix _A, Matrix _B){
            Matrix A = -_A;
            Matrix B = new Matrix(_B);

            for (int i = 0; i < _A.Rows; ++i){
                decimal a = -A[i, i];
                A[i, i] = B[i, 0];
                for (int j = 0; j < _A.Columns; ++j) {
                    A[i, j] /= a;
                }
            }
            Matrix xk = new Matrix(1, A.Columns);
            for (int i = 0; i < B.Rows; ++i) xk[0, i] = A[i, i];
            decimal dx = 1;
            while (dx >= epsilon){
                dx = 0;
                for (int i = 0; i < A.Rows; ++i){
                    decimal tmp = xk[0, i];
                    xk[0, i] = 0;
                    for (int j = 0; j < A.Columns; ++j){
                        if (i != j) xk[0, i] += A[i, j] * xk[0, j];
```

```
                    else xk[0, i] += A[i, j];
                }
                tmp -= xk[0, i];
                tmp = Math.Abs(tmp);
                if (tmp > dx) dx = tmp;
            }
        }
        for (int i = 0; i < xk.Columns; ++i)
            Console.WriteLine("x" + (i + 1) + " = " + xk[0, i] + "\n");
        Console.WriteLine("Подстановка:\n");
        for (int i = 0; i < _A.Rows; ++i) {
            decimal sum = 0;
            for (int j = 0; j < _A.Columns; ++j){
                sum += _A[i, j] * xk[0, j];
            }
            Console.WriteLine(i + ". " + sum + " = " + B[i, 0] + " => невязка = " + (sum - B[i,
0]) + "\n");
        }
    }
  }
}
```

Результаты расчетов программы:

```
Матрица A

11,97 1,89 1,53 1,08 -1,17
-1,17 -11,97 0,99 1,53 1,08
1,08 -1,17 -11,97 0,99 1,53
1,53 1,08 -1,17 -11,97 0,99
0,99 6,03 1,08 -1,17 -11,97



Матрица B

1,2
2,2
4,0
0,0
-1,2



Прямая иттерация

x1 = 0,1684350001602188017090893697

x2 = -0,2238863807017038346408769735

x3 = -0,2983736396163629134474061700

x4 = 0,0281547056331884038534626489

x5 = -0,0282764644449382861647712002

Подстановка:

1. 1,2000005692629848223865328003 = 1,2 => невязка = 0,0000005692629848223865328003

2. 2,1999992416099845271765756586 = 2,2 => невязка = -0,0000007583900154728234243414

3. 3,9999994997779948083239225193 = 4,0 => невязка = -0,0000005002220051916760774807

4. -0,0000001087913148634928465724 = 0,0 => невязка = -0,0000001087913148634928465724

5. -1,1999994824432486028319283706 = -1,2 => невязка = 0,0000005175567513971680716294
```

```
Метод Зейделя
x1 = 0,1684349190378035056515268339

x2 = -0,2238864186677022505268089688

x3 = -0,2983736685949804350530056530

x4 = 0,0281546951851184025237255262

x5 = -0,0282764486355878446860193389

Подстановка:

0. 1,1999994523537962965302747965 = 1,2 => невязка = -0,0000005476462037034697252035

1. 2,1999997633759314900915405334 = 2,2 => невязка = -0,0000002366240685099084594666

2. 3,9999998173047730429333718229 = 4,0 => невязка = -0,0000001826952269570666281771

3. -0,0000000992922512023582547105 = 0,0 => невязка = -0,0000000992922512023582547105

4. -1,2000000000000000000000000006 = -1,2 => невязка = -0,0000000000000000000000000006
```

Оценка:

Подсчет абсолютной погрешности приближенного решения:

Учитывая

$$\|\mathbf{r}\| \leqslant \|A^{-1}\| \, \|\eta\|.$$

Имеем для метода прямых итераций:

> $A1 := A^{-1};$

$A1 := [[0.0827848747184323, 0.0104730021831710, 0.0100039660371903, 0.0101270692738783, -0.00503054256012535], [-0.00576322314565676, -0.0895161915537739, -0.00785381461657969, -0.0116845059346439, -0.00948358504898261], [0.00930902705395988, 0.00333478488971422, -0.0826838096613464, -0.00444387262540964, -0.0115451676402053], [0.00947069626375699, -0.0106112738427274, 0.00771380336637938, -0.0826501899130857, -0.00773287044629371], [0.00385779657415361, -0.0428903596350071, -0.0113432113729386, 0.00262903655456179, -0.0890215185972554]]$

=

$$> b := \begin{bmatrix} 0.0000005692629848223865328003 \\ 0.0000007583900154728234243414 \\ 0.0000005002220051916760774807 \\ 0.0000001087913148634928465724 \\ 0.0000005175567513971680716294 \end{bmatrix};$$

$$b := \begin{bmatrix} 5.692629848223865328003\ 10^{-7} \\ 7.583900154728234243414\ 10^{-7} \\ 5.002220051916760774807\ 10^{-7} \\ 1.087913148634928465724\ 10^{-7} \\ 5.175567513971680716294\ 10^{-7} \end{bmatrix}$$

=

> $r \le norm(A1, 1) \cdot norm(b, 1);$

$$r \le 3.848850355\ 10^{-7}$$

=

Для метода Зейделя

$$> b := \begin{bmatrix} 0.0000005476462037034697252035 \\ 0.0000002366240685099084594666 \\ 0.0000001826952269570666281771 \\ 0.0000000992922512023582547105 \\ 0.0000000000000000000000000006 \end{bmatrix};$$

$$b := \begin{bmatrix} 5.476462037034697252035\ 10^{-7} \\ 2.366240685099084594666\ 10^{-7} \\ 1.826952269570666281771\ 10^{-7} \\ 9.929225120235825471 05\ 10^{-8} \\ 6.\ 10^{-28} \end{bmatrix}$$

> $r \le norm(A1, 1) \cdot norm(b, 1);$

$$r \le 1.672165243\ 10^{-7}$$

Относительная погрешность приближенного числа:

Учитывая формулу

$$\left|\frac{a - a^*}{a}\right| = \frac{\beta_{l+1} r^{-(l+1)} + \beta_{l+1} r^{-(l+2)} + \dots}{\beta_1 r^{-1} + \beta_2 r^{-2} + \dots} \le \frac{r^{-l}}{\beta_1 r^{-1}} \le r^{1-l}$$

где r-основание системы исчисления

И параметры используемого типа decimal

Ниже в таблице даны параметры стандартных форматов чисел с плавающей запятой. Здесь: **w** — ширина битового поля для представления порядка, **t** — ширина битового поля для представления мантиссы, **k** — полная ширина битовой строки.

Параметр	*Binary32*	*Binary64*	*Binary128*	*Decimal64*	*Decimal128*
Параметры формата					
b	2	2	2	10	10
p, цифры	24	53	113	16	34
emax	127	1023	16383	384	6144
Параметры кодирования					
BIAS	127	1023	16383	398	6176
w, биты	8	11	15	13	17
t, биты	23	52	112	50	110
k, биты	32	64	128	64	128

То относительная погрешность приближенного числа не превышает 2^(1 – 110) = 1,50463*10^(-36)

Вывод:

В ходе лабораторной работы был изучен метод прямых иттераций и метод Зейделядля решения системы линейных уравнений, составлены алгоритмы и написаны реализации соответствующих программ на языке C# для решения поставленной задачи. Проведена оценка погрешности. Решения были найдены с необходимой точностью.

СПИСОК ЛИТЕРАТУРЫ

1. ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ АЛГЕБРЫ: БАЗОВЫЕ ПОНЯТИЯ И АЛГОРИТМЫ / Б.В, Фалейчик, 2010
2. КОМПЕНСАЦИЯ ПОГРЕШНОСТЕЙ ПРИ ОПЕРАЦИЯХ С ЧИСЛАМИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ[Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/266023/