Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Отчет по лабораторной работе №1

«Решение СЛАУ с трехдиагональной матрицей методом прогонки»

по курсу

«Численные методы»

Студент группы ИУ9-62Б: Караник А.А.

Преподаватель: Домрачева А. Б.

Москва, 2024 г.

Цель работы

Целью данной работы является реализация программы для решения СЛАУ с трехдиагональной матрицей методом прогонки.

Постановка задачи

Дано: $A\overline{x} = \overline{d}$, где $A \in R^{n\times n}$ и $\overline{x}, \overline{d} \in R^n$, где A – трехдиагональная матрица.

Найти: Решение СЛАУ с помощью метода прогонки. Иными словами, найти $\overline{x}$ при заданных A и $\overline{d}$.

Теоретические сведения

Описание метода:

$A\overline{x} = \overline{d}$

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_1 & b_2 & c_2 & \ddots & \vdots \\ 0 & a_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & a_{n-1} & b_n \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

Имеем следующую систему:

$$\begin{cases} b_1x_1 + c_1x_2 = d_1 \\ a_1x_1 + b_2x_2 + c_2x_3 = d_2 \\ \vdots \\ a_{n-1}x_{n-1} + b_nx_n = d_n \end{cases}$$

$$x_1 = \frac{d_1}{b_1} - \frac{c_1}{b_1}x_2$$

Делаем замену:

$\alpha_1 = -\frac{c_1}{b_1}, \beta_1 = \frac{d_1}{b_1}$

$x_1 = \alpha_1x_2 + \beta_1$

Далее индуктивно получаем:

$x_i = \alpha_i x_{i+1} + \beta_i$

$i = \overline{n-1, \dots, 1}$

$$x_n = \beta_n$$

$$\begin{cases} \alpha_i = -\dfrac{c_i}{\alpha_{i-1}a_{i-1} + b_i} \\ \beta_i = \dfrac{d_i - a_{i-1}\beta_{i-1}}{\alpha_{i-1}a_{i-1} + b_i} \end{cases}, i = \overline{2, \dots, n}$$

Решение существует при выполнении следующих условий:

1) $|b_i| \geq |a_{i-1}| + |c_i|, i = \overline{2, \dots, n-1}$
2) $\dfrac{|b_i|}{|c_i|} \geq 1$
3) $\dfrac{|b_i|}{|a_{i-1}|} \geq 1$

Практическая реализация:

```
/**
 * Laboratory work: 1
 * Discipline: Numerical methods
 * Copyright (c) 2024, Andrey Karanik
*/

#include <iostream>
#include <cmath>
#include <iomanip>

int main(int argc, char *argv[]) {
    int n = 0;

    std::cout << "Enter size of matrix:" << std::endl;
    std::cin >> n;

    float *a = new float[n-1];
    float *b = new float[n];
    float *c = new float[n-1];
    float *d = new float[n];
    float *alpha = new float[n - 1];
    float *beta = new float[n - 1];
    float *x = new float[n];
    float *expectedX = new float[n];

    std::cout << "Enter lower diagonal:" << std::endl;
    for (int i = 0; i < n - 1; i++) {
        std::cin >> a[i];
    }

    std::cout << "Enter main diagonal:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> b[i];
    }

    std::cout << "Enter upper diagonal:" << std::endl;
    for (int i = 0; i< n - 1; i++) {
        std::cin >> c[i];
    }

    std::cout << "Enter d:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> d[i];
    }

    std::cout << "Enter expected x:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> expectedX[i];
    }

    for (int i = 1; i < n - 1; i++) {
        if (std::fabs(b[i]) < std::fabs(a[i - 1]) + std::fabs(c[i])) {
            std::cout << "The condition is not satisfied." << std::endl;
            break;
        }
        if (std::fabs(b[i]) / (std::fabs(c[i])) < 1) {
            std::cout << "The condition is not satisfied." << std::endl;
            break;
        }
```

```
        if (std::fabs(b[i]) / (std::fabs(a[i - 1])) < 1) {
            std::cout << "The condition is not satisfied." << std::endl;
            break;
        }
    }

    if (b[0] != 0) {
        alpha[0] = -c[0] / b[0];
    } else {
        std::cout << "ERROR: b[0] = 0";
        return 1;
    }

    beta[0] = d[0] / b[0];

    for (int i = 1; i < n - 1; i++) {
        alpha[i] = -(c[i]) / (alpha[i - 1] * a[i - 1] + b[i]);
        beta[i] = (d[i] - a[i - 1] * beta[i - 1]) / (alpha[i - 1] * a[i - 1] + b[i]);
    }
    beta[n - 1] = (d[n - 1] - a[n - 2] * beta[n - 2]) / (alpha[n - 2] * a[n - 2] + b[n - 1]);

    x[n - 1] = beta[n - 1];

    for (int i = n - 2; i >= 0; i--) {
        x[i] = alpha[i] * x[i + 1] + beta[i];
    }

    std::cout << std::endl;

    for (int i = 0; i < n; i++) {
        std::cout << "x[" << i << "] = " << std::fixed << std::setprecision(std::numeric_limits<float>::max_digits10) << x[i] << std::endl;
    }

    std::cout << std::endl;;


    for (int i = 0; i < n; i++) {
        std::cout << "error[" << i << "] = " << std::fixed << std::setprecision(std::numeric_limits<float>::max_digits10) << x[i] -
expectedX[i] << std::endl;
    }

    delete[] a;
    delete[] b;
    delete[] c;
    delete[] d;
    delete[] alpha;
    delete[] beta;
    delete[] x;
    delete[] expectedX;

    return 0;
}
```

Тестирование

1)

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix}, \overline{d} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}, \overline{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$\overline{x^*} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ – решение (приближенное)

$\overline{\varepsilon} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ – математическая погрешность

```
Enter size of matrix:
4
Enter lower diagonal:
1
1
1
Enter main diagonal:
4
4
4
4
Enter upper diagonal:
1
1
1
Enter d:
5
6
6
5
Enter expected x:
1
1
1
1

x[0] = 1.000000000
x[1] = 1.000000000
x[2] = 1.000000000
x[3] = 1.000000000

error[0] = 0.000000000
error[1] = 0.000000000
error[2] = 0.000000000
error[3] = 0.000000000
```

2)

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & -1 & -1 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \overline{d} = \begin{pmatrix} 5 \\ 3 \\ -3 \\ 7 \end{pmatrix}, \overline{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$\overline{x^*} = \begin{pmatrix} 1.000000477 \\ 1.999999762 \\ 3.000000238 \\ 4.000000477 \end{pmatrix}$$ – решение (приближенное)

$$\overline{\varepsilon} = \begin{pmatrix} 0.000000477 \\ -0.000000238 \\ 0.000000238 \\ 0.000000477 \end{pmatrix}$$ – математическая погрешность

```
Enter size of matrix:
4
Enter lower diagonal:
2
1
1
Enter main diagonal:
1
-1
-1
1
Enter upper diagonal:
2
-1
1
Enter d:
5
-3
3
7
Enter expected x:
```

```
1
2
3
4
The condition is not satisfied.

x[0] = 1.000000477
x[1] = 1.999999762
x[2] = 3.000000238
x[3] = 4.000000477

error[0] = 0.000000477
error[1] = -0.000000238
error[2] = 0.000000238
error[3] = 0.000000477
```

Вывод

Программа реализована и работает корректно, решая СЛАУ методом прогонки, однако имеет место быть математическая погрешность.