**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

|  |
| --- |
|  |

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ЮЖНО-РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (НПИ) им. М.И. Платова»**

|  |
| --- |
|  |

**ФАКУЛЬТЕТ** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**НАПРАВЛЕНИЕ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ОТЧЕТ**

**по лабораторным работам**

**по дисциплине: \_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил студент \_\_\_\_\_\_курса, группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Фамилия, имя, отчество

Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность, звание Фамилия, имя, отчество

Работа принята «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

Новочеркасск 2024 г.

**РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ: ПРЯМЫЕ МЕТОДЫ**

**Цели**:

1. Реализовать метод Гаусса решения системы линейных алгебраических уравнений.

2. Реализовать метод прогонки решения системы линейных алгебраических уравнений.

**1 Реализация метода Гаусса**

**Логика решения задачи:**

**Прямой ход**:

Приводим матрицу A к верхнетреугольному виду.

Используем перестановку строк для улучшения численной стабильности.

Убираем элементы под главной диагональю путём вычитания строк.

**Обратный ход**:

После приведения к треугольной форме, решаем уравнения, начиная с нижней строки, поэтапно вычисляя значения неизвестных.

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using PuzanovVE.NM;

namespace PuzanovVE.NM

{

//14 вариант

public class GaousseSolver : SlauSolver

{

/// <summary>

/// Метод Гаусса решения СЛАУ.

/// </summary>

public GaousseSolver() { }

/// <summary>

/// Решает СЛАУ.

/// </summary>

/// <param name="a">Матрица коэффициентов.</param>

/// <param name="b">Правая часть.</param>

/// <returns>Вектор неизвестных – решение

/// СЛАУ.</returns>

public override double[] Solve(double[][] a, double[] b)

{

// Решение СЛАУ методом Гаусса

// ...

double akk, aik;

int n = b.Length;

for (int k = 0; k < n; k++)

{

akk = a[k][k];

if (akk != 0)

{

b[k] /= akk;

for (int j = k; j < n; j++)

{

a[k][j] /= akk;

}

}

for (int i = k + 1; i < n; i++)

{

aik = a[i][k];

if (akk != 0)

{

b[i] /= aik;

b[i] -= b[k];

for (int j = k; j < n; j++)

{

a[i][j] /= aik;

a[i][j] -= a[k][j];

}

}

}

}

for (int i = n - 2; i >= 0; i--)

{

for (int j = i + 1; j < n; j++)

{

b[i] -= a[i][j] \* b[j];

}

}

return b;

}

internal class Program

{

public static double[] MultiplyMatrix(double[][] matrix, double[] vector)

{

int rows = 3; // Количество строк в матрице

int cols = 3; // Количество столбцов в матрице

double[] result = new double[rows]; // Результирующий вектор

// Умножение

for (int i = 0; i < rows; i++)

{

result[i] = 0; // Инициализация результата

for (int j = 0; j < cols; j++)

{

result[i] += matrix[i][j] \* vector[j];

}

}

return result; // Возвращаем результирующий вектор

}

static void Main(string[] args)

{

Console.WriteLine("\t{ 16, -4, -28 },");

Console.WriteLine("A = \t{ -4, 10, 1 },");

Console.WriteLine("\t{ -28, 1, 69 },\n");

Console.WriteLine("B = \t{ 52, -25, -99 },\n");

double[][] A = new double[][]

{

new double[] { 16, -4, -28 },

new double[] { -4, 10, 1 },

new double[] { -28, 1, 69 }

};

double[] B = new double[]

{

52,

-25,

-99

};

GaousseSolver solver = new GaousseSolver();

double[] result = solver.Solve(A, B);

double[][] A\_ = new double[][]

{

new double[] { 16, -4, -28 },

new double[] { -4, 10, 1 },

new double[] { -28, 1, 69 }

};

double[] check = MultiplyMatrix(A\_, result);

Console.WriteLine("Решение СЛАУ:\n");

Console.Write("{");

foreach (var value in result)

{

Console.Write(value);

Console.Write("\t");

}

Console.Write("}\n");

Console.WriteLine("Проверка решения СЛАУ:\n");

Console.Write("{");

foreach (var value in check)

{

Console.Write(value);

Console.Write("\t");

}

Console.Write("}");

Console.ReadLine();

}

}

}

}

**Исходные данные:**

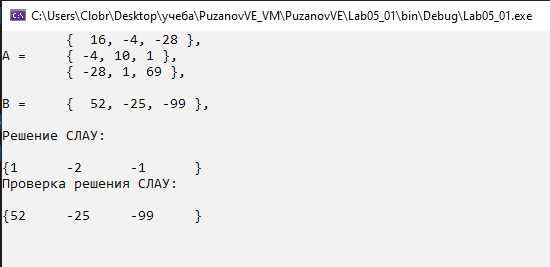
Матрица А:

|  |  |  |
| --- | --- | --- |
| 16 | -4 | 28 |
| -4 | 10 | 1 |
| -28 | 1 | 69 |

Вектор b

|  |
| --- |
| 52 |
| -25 |
| 99 |

**Пример работы программы:**



**2 Реализация метода прогонки**

**Логика решения задачи:**

Система уравнений имеет вид:

aixi−1+bixi+cixi+1=fi

где:

a, b, и c — коэффициенты трехдиагональной матрицы,

f — вектор правых частей.

Метод прогонки включает два шага: прямой ход и обратный ход. Прямой ход заключается в вычислении вспомогательных коэффициентов, а обратный ход — в нахождении решения системы.

**Прямой ход**:

Вычисляются коэффициенты α и β, которые позволяют решить систему.α и β рассчитываются рекурсивно, начиная с первого элемента.

**Обратный ход**:

После вычисления всех коэффициентов, система решается с конца, начиная с последнего уравнения.

**Код программы:**

using System;

namespace PuzanovVE.NM

{

public class ThomasSolver

{

/// <summary>

/// Решает систему линейных уравнений Ax = B, где A - трехдиагональная матрица.

/// </summary>

/// <param name="a">Элементы поддиагонали</param>

/// <param name="b">Элементы главной диагонали</param>

/// <param name="c">Элементы наддиагонали</param>

/// <param name="d">Правая часть</param>

/// <returns>Вектор решений x</returns>

public double[] Solve(double[] a, double[] b, double[] c, double[] d)

{

int n = d.Length; // Количество уравнений

double[] x = new double[n]; // Вектор решений

double[] cp = new double[n - 1]; // Модифицированная наддиагональ

double[] dp = new double[n]; // Модифицированная правая часть

// Прямой ход

cp[0] = c[0] / b[0]; // Первый элемент модифицированной наддиагонали

dp[0] = d[0] / b[0]; // Первый элемент модифицированной правой части

for (int i = 1; i < n; i++)

{

double denominator = b[i] - a[i - 1] \* cp[i - 1]; // Находим знаменатель

if (i < n - 1) cp[i] = c[i] / denominator; // Модифицированная наддиагональ

dp[i] = (d[i] - a[i - 1] \* dp[i - 1]) / denominator; // Модифицированная правая часть

}

// Обратный ход

x[n - 1] = dp[n - 1]; // Последний элемент решения

for (int i = n - 2; i >= 0; i--)

{

x[i] = dp[i] - cp[i] \* x[i + 1]; // Рассчитываем остальные элементы решения

}

return x; // Возвращаем вектор решений

}

/// <summary>

/// Умножает матрицу на вектор.

/// </summary>

/// <param name="matrix">Двумерный массив, представляющий матрицу.</param>

/// <param name="vector">Одномерный массив, представляющий вектор.</param>

/// <returns>Результат умножения матрицы на вектор.</returns>

public static double[] MultiplyMatrix(double[][] matrix, double[] vector)

{

int rows = 5; // Количество строк в матрице

int cols = 5; // Количество столбцов в матрице

double[] result = new double[rows]; // Результирующий вектор

// Проверка совместимости

if (cols != vector.Length)

{

throw new ArgumentException("Количество столбцов матрицы должно совпадать с размерностью вектора.");

}

// Умножение

for (int i = 0; i < rows; i++)

{

result[i] = 0; // Инициализация результата

for (int j = 0; j < cols; j++)

{

result[i] += matrix[i][j] \* vector[j];

}

}

return result; // Возвращаем результирующий вектор

}

}

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("\t{ -4, -2, 0, 0, 0 },");

Console.WriteLine("\t{ -2, -4, -1, 0, 0 },");

Console.WriteLine("A = \t{ 0, -1, -4, -1, 0 },");

Console.WriteLine("\t{ 0, 0, -1, -4, -2 },");

Console.WriteLine("\t{ 0, 0, 0, -2, -4 }\n");

Console.WriteLine("B = \t{ 2, -5, -5, -3, 4 },\n");

// Определение матрицы коэффициентов (трехдиагональная матрица)

double[][] A = new double[][]

{

new double[] { -4, -2, 0, 0, 0 },

new double[] { -2, -4, -1, 0, 0 },

new double[] { 0, -1, -4, -1, 0 },

new double[] { 0, 0, -1, -4, -2 },

new double[] { 0, 0, 0, -2, -4 }

};

// Определение правой части

double[] B = new double[] { 2, -5, -5, -3, 4 };

// Создание экземпляра ThomasSolver

ThomasSolver solver = new ThomasSolver();

// Подготовка массивов для метода

int n = B.Length; // Количество уравнений

double[] a = new double[n - 1]; // Поддиагональ

double[] b = new double[n]; // Главная диагональ

double[] c = new double[n - 1]; // Наддиагональ

// Заполнение массивов a, b, c

for (int i = 0; i < n; i++)

{

b[i] = A[i][i]; // Заполнение главной диагонали

if (i < n - 1)

{

c[i] = A[i][i + 1]; // Заполнение наддиагонали

}

if (i > 0)

{

a[i - 1] = A[i][i - 1]; // Заполнение поддиагонали

}

}

// Вывод информации о массивах

Console.WriteLine("Поддиагональ (a): \t" + string.Join(", ", a));

Console.WriteLine("Главная диагональ (b): \t" + string.Join(", ", b));

Console.WriteLine("Наддиагональ (c): \t" + string.Join(", ", c));

Console.WriteLine("Правая часть (B): \t" + string.Join(", ", B));

// Вызов метода Solve

double[] result = solver.Solve(a, b, c, B);

// Вывод результата

Console.WriteLine("Решение СЛАУ:");

foreach (var value in result)

{

Console.WriteLine(value);

}

// Вызов метода MultiplyMatrix

double[] check = ThomasSolver.MultiplyMatrix(A, result);

// Вывод результата

Console.WriteLine("Проверка решения СЛАУ:\t");

Console.WriteLine("{ -4, -2, 0, 0, 0 },\t\t" + result[0] + "\t=" + check[0]);

Console.WriteLine("{ -2, -4, -1, 0, 0 },\t\t" + result[1] + "\t=" + check[1]);

Console.WriteLine("{ 0, -1, -4, -1, 0 }, \* \t" + result[2] + "\t\t\t=" + check[2]);

Console.WriteLine("{ 0, 0, -1, -4, -2 },\t\t" + result[3] + "\t=" + check[3]);

Console.WriteLine("{ 0, 0, 0, -2, -4 }\t\t" + result[4] + "\t=" + check[4]);

Console.WriteLine("\n=\n");

Console.ReadLine();

}

}

**Исходные данные:**

a [-2, -1, -1, -2], поддиагональ  
b [-4, -4, -4, -4, -4], главная диагональ  
c [-2, -1, -1, -2], наддиагональ  
f [2, -5, -5, -3, 4], правая часть

**Пример работы программы:**

