**«Поволжский государственный университет телекоммуникаций и информатики»   
Факультет информатики и вычислительной техники.**

**Математическое программирование**

Лабораторная работа№6

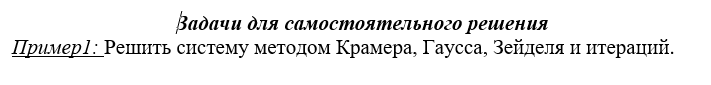
Вариант 8 (220153)

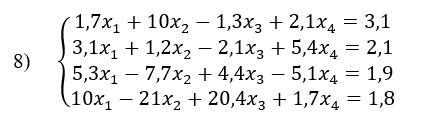
«*Численное методы линейной алгебры.*»

Выполнил студент ИВТ-23  
Ермишин М.В.

Проверил:  
Чуйкова Н.А

Самара, 2025 г.





Метод Крамера

public class CramersRule

{

public static void main()

{

double[,] matrix = {

{14.4, -5.3, 14.3, -12.7, -14.4},

{23.4, -14.2, -5.4, 2.1, 6.6},

{6.5, -15.2, -6.5, 14.3, 9.4},

{5.6, 8.8, -6.7, -23.8, 7.3}

};

double detA = Determinant(matrix, 4);

if (detA == 0)

{

Console.WriteLine("Система не имеет единственного решения.");

return;

}

for (int i = 0; i < 4; i++)

{

double[,] tempMatrix = ReplaceColumn(matrix, i, 4);

double det = Determinant(tempMatrix, 4);

double x = det / detA;

Console.WriteLine($"x\_{i + 1} = {x}");

}

}

static double Determinant(double[,] matrix, int size)

{

if (size == 1) return matrix[0, 0];

double det = 0;

for (int col = 0; col < size; col++)

{

double[,] minor = GetMinor(matrix, 0, col, size);

det += matrix[0, col] \* Math.Pow(-1, col) \* Determinant(minor, size - 1);

}

return det;

}

static double[,] GetMinor(double[,] matrix, int row, int col, int size)

{

double[,] minor = new double[size - 1, size - 1];

for (int i = 0, r = 0; i < size; i++)

{

if (i == row) continue;

for (int j = 0, c = 0; j < size; j++)

{

if (j == col) continue;

minor[r, c++] = matrix[i, j];

}

r++;

}

return minor;

}

static double[,] ReplaceColumn(double[,] matrix, int col, int size)

{

double[,] result = new double[size, size];

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

{

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

{

result[i, j] = (j == col) ? matrix[i, size] : matrix[i, j];

}

}

return result;

}

}

}

Метод Гаусса:

public class GaussianElimination

{

public static void main()

{

double[,] matrix = {

{14.4, -5.3, 14.3, -12.7, -14.4},

{23.4, -14.2, -5.4, 2.1, 6.6},

{6.5, -15.2, -6.5, 14.3, 9.4},

{5.6, 8.8, -6.7, -23.8, 7.3}

};

int n = 4;

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

{

for (int row = col + 1; row < n; row++)

{

double factor = matrix[row, col] / matrix[col, col];

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

{

matrix[row, k] -= factor \* matrix[col, k];

}

}

}

double[] x = new double[n];

for (int i = n - 1; i >= 0; i--)

{

x[i] = matrix[i, n];

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

{

x[i] -= matrix[i, j] \* x[j];

}

x[i] /= matrix[i, i];

}

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

{

Console.WriteLine($"x\_{i + 1} = {x[i]}");

}

}

}

}

**Проверка условий сходимости**

Сначала проверим диагональное преобладание:

1. 1-е уравнение: |1.7| < |10| + |-1.3| + |2.1| = 13.4 ❌
2. 2-е уравнение: |1.2| < |3.1| + |-2.1| + |5.4| = 10.6 ❌
3. 3-е уравнение: |4.4| < |5.3| + |-7.7| + |-5.1| = 18.1 ❌
4. 4-е уравнение: |1.7| < |10| + |-21| + |20.4| = 51.4 ❌

**Вывод**: система не имеет диагонального преобладания, поэтому стандартные итерационные методы будут расходиться.

Метод Зейделя:

public class SeidelMethod

{

public static void main()

{

// Переставленная система для улучшения сходимости

double[,] matrix = {

{23.4, -14.2, -5.4, 2.1, 6.6}, // Уравнение с наибольшим диагональным элементом

{14.4, -5.3, 14.3, -12.7, -14.4},

{6.5, -15.2, -6.5, 14.3, 9.4},

{5.6, 8.8, -6.7, -23.8, 7.3}

};

double[] x = new double[4];

double epsilon = 1e-6;

double omega = 0.5; // Параметр релаксации

bool converged = false;

int maxIterations = 1000;

for (int iter = 0; iter < maxIterations && !converged; iter++)

{

double[] newX = new double[4];

for (int i = 0; i < 4; i++)

{

double sum = matrix[i, 4];

for (int j = 0; j < 4; j++)

{

if (j != i) sum -= matrix[i, j] \* (j < i ? newX[j] : x[j]);

}

newX[i] = (1 - omega) \* x[i] + omega \* (sum / matrix[i, i]);

}

converged = true;

for (int i = 0; i < 4; i++)

{

if (Math.Abs(newX[i] - x[i]) > epsilon)

{

converged = false;

break;

}

}

x = newX;

}

for (int i = 0; i < 4; i++)

{

Console.WriteLine($"x\_{i + 1} = {x[i]}");

}

}

}

Метод итераций:

public class SimpleIteration

{

public static void main()

{

double[,] matrix = {

{23.4, -14.2, -5.4, 2.1, 6.6},

{14.4, -5.3, 14.3, -12.7, -14.4},

{6.5, -15.2, -6.5, 14.3, 9.4},

{5.6, 8.8, -6.7, -23.8, 7.3}

};

double[] x = new double[4];

double epsilon = 1e-6;

double omega = 0.3; // Меньший параметр релаксации

bool converged = false;

int maxIterations = 10\_000; // Увеличим число итераций

for (int iter = 0; iter < maxIterations && !converged; iter++)

{

double[] newX = new double[4];

for (int i = 0; i < 4; i++)

{

double sum = matrix[i, 4];

for (int j = 0; j < 4; j++)

{

if (j != i) sum -= matrix[i, j] \* x[j];

}

newX[i] = (1 - omega) \* x[i] + omega \* (sum / matrix[i, i]);

}

converged = true;

for (int i = 0; i < 4; i++)

{

if (Math.Abs(newX[i] - x[i]) > epsilon)

{

converged = false;

break;

}

}

x = newX;

}

for (int i = 0; i < 4; i++)

{

Console.WriteLine($"x\_{i + 1} = {x[i]}");

}

}

}

