

Национальный исследовательский университет ИТМО

Факультет инфокоммуникационных технологий

**Лабораторная работа 4**

Выполнил: Орел

Даниил Максимович

Группа № K3221

Проверил: Иванов С.Е.

Санкт-Петербург

2020

**Цель работы:**

* Реализовать алгоритм решения системы уравнений методом Гаусса

**Ход работы:**

1. В программе был реализован класс **EquationsSystem,** в котором была реализована преобразование 2 подающихся на вход матриц (матрицы коэффициентов и матрицы свободных членов) в расширенную матрицу, а также реализован основной метод **Solve**, позволяющий решить систему методом Гаусса.

class EquationsSystem

{

private double[] vector;

public int Rows { get; set; }

public int Columns { get; set; }

public double[,] Matrix { get; set; }

public EquationsSystem(double[,] matrix, double[] vector)

{

Rows = matrix.GetLength(0);

Columns = matrix.GetLength(1) + 1;

Matrix = new double[Rows, Columns];

for(int i = 0; i < Matrix.GetLength(0); i++)

{

for(int j = 0; j < Columns - 1; j++)

{

Matrix[i, j] = matrix[i, j];

}

Matrix[i, Columns - 1] = vector[i];

}

}

private void PrintSystem()

{

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

{

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

{

if (j != Columns - 1)

Console.Write($"{Matrix[i, j]:0.000}\t");

else Console.Write($"|\t{Matrix[i, j]:0.000}");

}

Console.WriteLine();

}

Console.WriteLine();

}

private void PrintVector()

{

Console.WriteLine("Ответ: ");

for (int i = 0; i < vector.Length; i++)

{

Console.WriteLine($"x{i+1} = {vector[i]:0.00}");

}

Console.WriteLine();

}

private bool SelectLeading(int n)

{

//Найдем номер строки, с наибольшим

//элементом в столбце n

int iMax = n;

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

if (Math.Abs(Matrix[iMax, n]) < Math.Abs(Matrix[i, n]))

iMax = i;

// Переставить строки iMax и n

if (iMax != n)

{

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

{

double t = Matrix[n, i];

Matrix[n, i] = Matrix[iMax, i];

Matrix[iMax, i] = t;

}

return true;

}

return false;

}

private void SubtractRow(int k)

{

double m = Matrix[k, k];

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

{

double t = Matrix[i, k] / m;

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

{

Matrix[i, j] = Matrix[i, j] - Matrix[k, j] \* t;

}

}

}

private bool ToTriangleMatrix()

{

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

{

SelectLeading(i - 1);

if (Math.Abs(Matrix[i - 1, i - 1]) != 0)

{

SubtractRow(i - 1);

}

else return false;

}

return true;

}

private int Rank(bool isExtendedMatrix)

{

int rang = 0;

int q = 1;

int rows, columns;

rows = Rows;

if (isExtendedMatrix)

columns = Columns;

else columns = Columns - 1;

while (q <= MinValue(rows, columns))

{

double[,] matbv = new double[q, q];

for (int i = 0; i < (rows - (q - 1)); i++)

{

for (int j = 0; j < (columns - (q - 1)); j++)

{

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

{

for (int c = 0; c < q; c++)

{

matbv[k, c] = Matrix[i + k, j + c];

}

}

if (CalculateDeterminant(matbv, matbv.GetLength(0)) != 0)

{

rang = q;

}

}

}

q++;

}

return rang;

}

private int MinValue(int a, int b)

{

if (a >= b)

return b;

else

return a;

}

private double CalculateDeterminant(double[,] matrix, int rows)

{

double det = 0;

int k = 1;

double[,] new\_matrix = new double[rows, rows];

if (rows == 1)

return matrix[0, 0];

else if (rows == 2)

{

det = matrix[0, 0] \* matrix[1, 1] - matrix[1, 0] \* matrix[0, 1];

return det;

}

else

{

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

{

det += k \* matrix[i, 0] \* CalculateDeterminant(GetMatr(matrix, new\_matrix, i, 0, rows), rows - 1);

k = -k;

}

return det;

}

}

//Получение матрицы путем вычеркивание i-ой строки и j-ого столбца

private double[,] GetMatr(double[,] matrix, double[,] p, int i, int j, int m)

{

int ki, kj, di, dj;

di = 0;

for (ki = 0; ki < m - 1; ki++)

{ // проверка индекса строки

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < m - 1; kj++)

{ // проверка индекса столбца

if (kj == j) dj = 1;

p[ki, kj] = matrix[ki + di, kj + dj];

}

}

return p;

}

private int[] FindBaseVariables(int r)

{

int rows, columns;

int[] baseVar = new int[r];

rows = Rows;

columns = Columns - 1;

double[,] matbv = new double[r, r];

for (int i = 0; i < (rows - (r - 1)); i++)

{

for (int j = 0; j < (columns - (r - 1)); j++)

{

if (CalculateDeterminant(matbv, matbv.GetLength(0)) == 0)

{

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

{

for (int c = 0; c < r; c++)

{

matbv[k, c] = Matrix[i + k, j + c];

baseVar[c] = j+c;

}

}

}

else break;

}

}

return baseVar;

}

public void Solve()

{

PrintSystem();

if (Rank(true) == Rank(false))

{

int rank = Rank(true);

vector = new double[Columns - 1];

ToTriangleMatrix();

PrintSystem();

Console.Write("Система совместна и ");

if (rank == Columns - 1)

{

SolveDefinedSystem();

}

else

{

SolveIndefinedSystem(rank);

}

PrintVector();

}

else

{

Console.WriteLine($"Ранг матрицы системы не равен рангу расширенной матрицы: {Rank(true)} != {Rank(false)}.\nСистема не имеет решений.\n");

}

}

private void SolveIndefinedSystem(int rank)

{

Console.WriteLine("имеет бесконечное множество решений");

int[] baseVar = FindBaseVariables(rank);

bool[] free = new bool[vector.Length];

FindBaseAndFreeVariables(baseVar, free);

SetValueForFreeVariables(free);

FindBaseVariables(free);

}

private void FindBaseVariables(bool[] free)

{

int new\_rows = CountNonZeroRows();

double[] sum = new double[new\_rows];

for (int n = new\_rows - 1; n >= 0; n--)

{

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

sum[n] += vector[i] \* Matrix[n, i];

}

for (int n = new\_rows - 1; n >= 0; n--)

{

if (!free[n])

{

if (Matrix[n, n] != 0)

vector[n] = (Matrix[n, Columns - 1] - sum[n]) / Matrix[n, n];

else

{

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

{

sum[i] = 0;

for (int j = 0; j < Columns - 1; j++)

sum[i] += vector[j] \* Matrix[i, j];

vector[n] = (Matrix[i, Columns - 1] - sum[i]) / Matrix[i, n];

if (!Double.IsNaN(vector[n]))

{

break;

}

}

}

}

}

}

private int CountNonZeroRows()

{

int new\_rows = Rows;

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

{

int count = 0;

for (int j = 0; j < Columns - 1; j++)

{

if (Matrix[i, j] == 0) count++;

}

if (count == Columns - 1) new\_rows--;

}

return new\_rows;

}

private void SetValueForFreeVariables(bool[] free)

{

Console.WriteLine("Введите значения свободных переменных для получения частного решения: ");

for (int i = 0; i < vector.Length; i++)

{

if (free[i])

{

Console.Write($"x{i + 1} = ");

vector[i] = double.Parse(Console.ReadLine());

}

}

}

private void FindBaseAndFreeVariables(int[] baseVar, bool[] free)

{

Console.Write("Базисные переменные: ");

for (int i = 0; i < baseVar.Length; i++)

Console.Write($"x{baseVar[i] + 1} ");

Console.WriteLine();

Console.Write("Свободные переменные: ");

for (int i = 0; i < vector.Length; i++)

{

int count = 0;

for (int j = 0; j < baseVar.Length; j++)

{

if (i != baseVar[j])

{

count++;

}

}

if (count == baseVar.Length)

free[i] = true;

}

for (int i = 0; i < vector.Length; i++)

{

if (free[i])

Console.Write($"x{i + 1} ");

}

Console.WriteLine();

}

private void SolveDefinedSystem()

{

Console.WriteLine("имеет 1 единственное решение");

int nb = Columns - 1;

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

{

double sum = 0;

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

sum += vector[i] \* Matrix[n, i];

vector[n] = (Matrix[n, nb] - sum) / Matrix[n, n];

}

}

}

1. Примеры уравнений объявлены в методе **Main** класса **Program**:

class Program

{

//Example 1

double[,] matrix1 = new double[,] { { 4, -3, 2, -1 },

{ 3, -2, 1, -3 },

{ 5, -3, 1, -8} };

double[] vector1 = new double[] { 8, 7, 1 };

EquationsSystem system1 = new EquationsSystem(matrix1, vector1);

system1.Solve();

//Example 2

double[,] matrix2 = new double[,] { { 1, 2, 2, 3 },

{ 6, -3, -3, -1 },

{ -7, 1, 1, -2},

{ -3, 9, 9, 10}};

double[] vector2 = new double[] { 1, -9, 8 , 12 };

EquationsSystem system2 = new EquationsSystem(matrix2, vector2);

system2.Solve();

//Example 3

double[,] matrix3 = new double[,] { {1, 1, -1, 1 },

{2, -1, 3, -2},

{1, 0, -1, 2},

{3, -1, 1, -1} };

double[] vector3 = new double[] { 4, 1, 6, 0};

EquationsSystem system3 = new EquationsSystem(matrix3, vector3 );

system3.Solve();

//Example 4

double[,] matrix4 = new double[,] { {2, 3, -1, 1 },

{8, 12, -9, 8},

{4, 6, 3, -2},

{2, 3, 9, -7} };

double[] vector4 = new double[] { 1, 3, 3, 3 };

EquationsSystem system4 = new EquationsSystem(matrix4, vector4);

system4.Solve();

Console.ReadKey();

}

1. Результат работы программы:

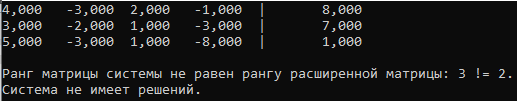


Рисунок 1 – Пример решения несовместной системы (Example 1)

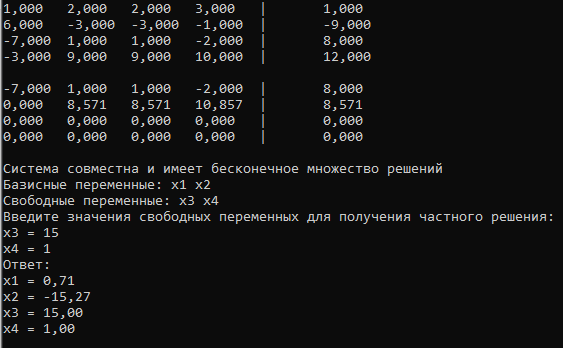


Рисунок 2 – Пример решения совместной системы, имеющей бесконечное количество решений (Example 2)

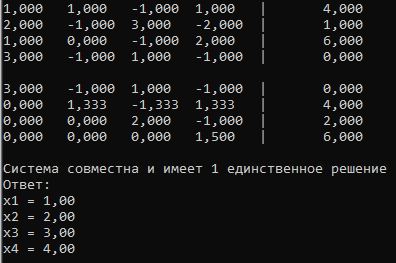


Рисунок 3 – Пример решения совместной определенной системы (Example 3)

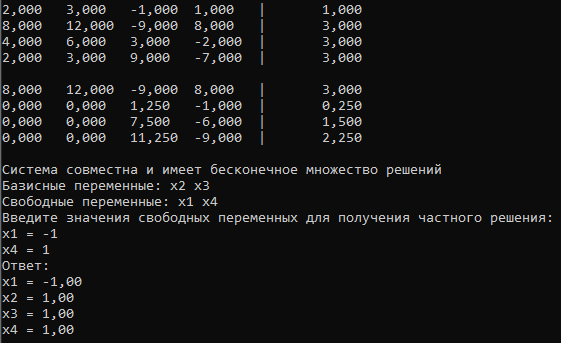


Рисунок 4 – Пример решения совместной системы, имеющей бесконечное количество решений (Example 4)

**Вывод:**

В ходе выполнения данной лабораторной работы был реализован алгоритм решения системы линейных уравнений методом Гаусса.