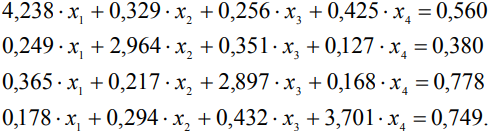


**Решим систему уравнений методом Гаусса**

****

На первом этапе заменим второе, третье, четвертое уравнения на уравнения, получающиеся сложением этих уравнений с первым, умноженным соответственно на , , , таким образом мы избавимся от всех х1 и система уравнений примет вид:

4,238 ∙ х1 + 0,329 ∙ х2 + 0,256 ∙ х3 + 0,425 ∙ х4 = 0,560

2,945 ∙ x2 + 0,336 ∙ x3 + 0,102 ∙ x4 = 0,347

0,189 ∙ x2 + 2,875 ∙ x3 + 0,131 ∙ x4 = 0,73

0,264 ∙ x2 + 0,421 ∙ x3 + 3,683 ∙ x4 = 0,725

На втором этапе проделываем такие же операции с полученной подсистемой, т. е. исключаем х2 из третьего и четвертого уравнения:

4,238 ∙ х1 + 0,329 ∙ х2 + 0,256 ∙ х3 + 0,425 ∙ х4 = 0,560

2,945 ∙ x2 + 0,336 ∙ x3 + 0,102 ∙ x4 = 0,347

2,853 ∙ x3 + 0,124 ∙ x4 = 0,708

0,39 ∙ x3 + 3,674 ∙ x4 = 0,694

В третьем шаге исключаем х3 из четвертого уравнения:

4,238 ∙ х1 + 0,329 ∙ х2 + 0,256 ∙ х3 + 0,425 ∙ х4 = 0,560

2,945 ∙ x2 + 0,336 ∙ x3 + 0,102 ∙ x4 = 0,347

2,853 ∙ x3 + 0,124 ∙ x4 = 0,708

3,08 ∙ x4 = 0,59

Таким образом находим неизвестные:

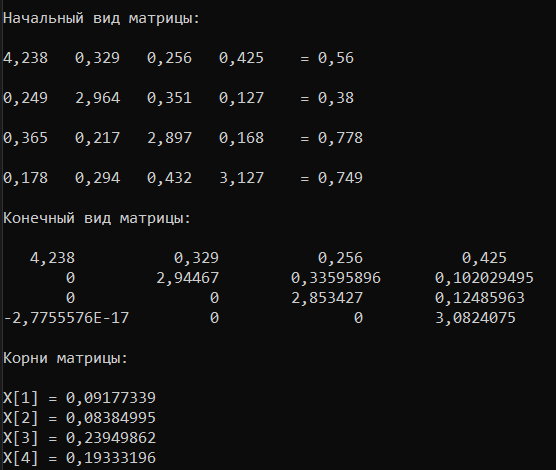
х4 = = 0,192

х3 = = 0,239

х2 = = 0,0839

х1 = = 0,092

**Скриншот работы:**

****

**Код:**

namespace Gauss

{

internal class Program

{

static void Main()

{

Console.WriteLine();

double[,] m = new double[4, 4];//матрица чисел

double[] f = new double[4];//массив свободных

double Multi1, Multi2;

double[] Result = new double[4];// массив результатов

m[0, 0] = 4.238;

m[0, 1] = 0.329;

m[0, 2] = 0.256;

m[0, 3] = 0.425;

f[0] = 0.56;

m[1, 0] = 0.249;

m[1, 1] = 2.964;

m[1, 2] = 0.351;

m[1, 3] = 0.127;

f[1] = 0.38;

m[2, 0] = 0.365;

m[2, 1] = 0.217;

m[2, 2] = 2.897;

m[2, 3] = 0.168;

f[2] = 0.778;

m[3, 0] = 0.178;

m[3, 1] = 0.294;

m[3, 2] = 0.432;

m[3, 3] = 3.127;

f[3] = 0.749;

Console.WriteLine("Начальный вид матрицы:");

Console.WriteLine();

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

{

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

{

Console.Write(m[i, j] + "\t");

}

Console.WriteLine($" = {f[i]}");

Console.WriteLine();

}

for (int k = 0; k < 4; k++)//начинаем обнулять строки поэлементно

{

for (int j = k + 1; j < 4; j++)

{

Multi1 = m[j, k] / m[k, k];//берум элемент 1,1 и делим на элемент под ним и т.д.

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

{

m[j, i] = m[j, i] - Multi1 \* m[k, i];

}

f[j] = f[j] - Multi1 \* f[k];

}

}

for (int k = 3; k >= 0; k--)

{

Multi1 = 0;

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

{

Multi2 = m[k, j] \* Result[j];

Multi1 += Multi2;

}

Result[k] = (f[k] - Multi1) / m[k, k];

}

Console.WriteLine("Конечный вид матрицы:");

Console.WriteLine();

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

{

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

{

Console.Write("{0, 9}", (float)m[i, j] + "\t");

}

Console.WriteLine();

}

Console.WriteLine();

Console.WriteLine("Корни матрицы:");

Console.WriteLine();

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

{

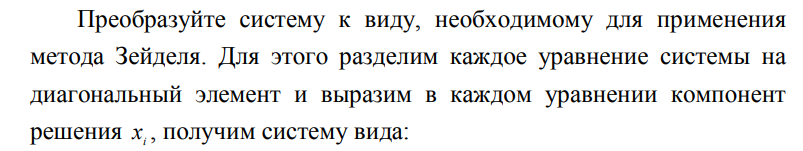
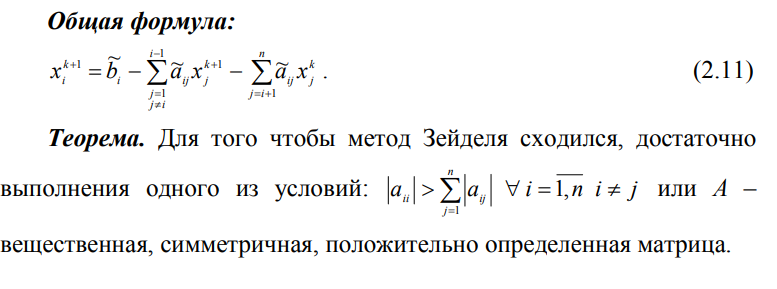
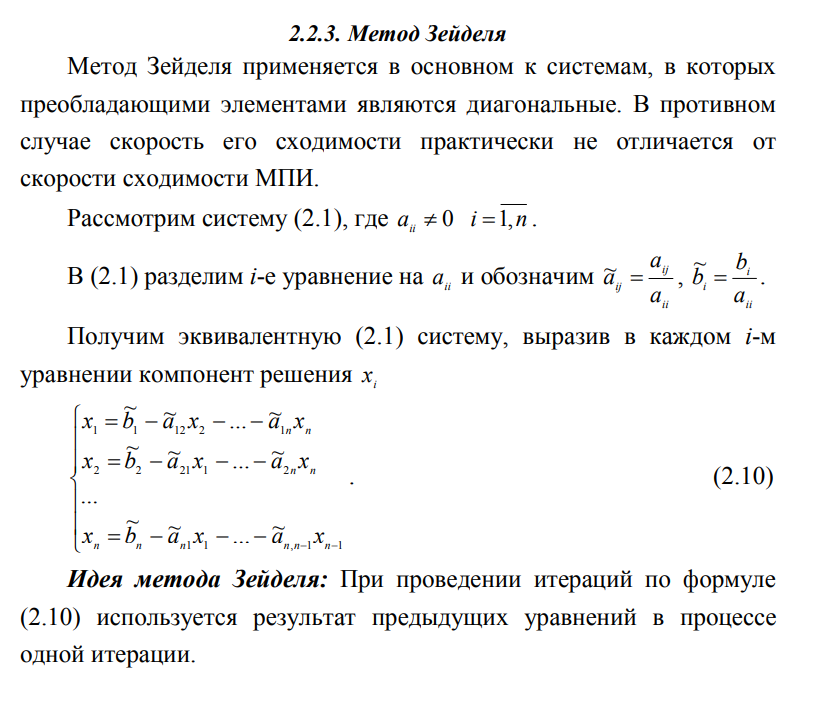
Console.WriteLine($"X[{i + 1}] = {(float)Result[i]}");

}

}

}

}



x1= 0,1321 - 0,0776 x2 - 0,06 x3 - 0,1 x4

x2= 0,128 - 0,084 x1 - 0,118 x3 - 0,0428 x4

x3= 0,126 – 0,075 x1 – 0,075 x2 – 0,058 x4

x4 = 0,24 – 0,057 x1 – 0,094 x2 – 0,138

Проверим условие сходимости метода Зейделя:

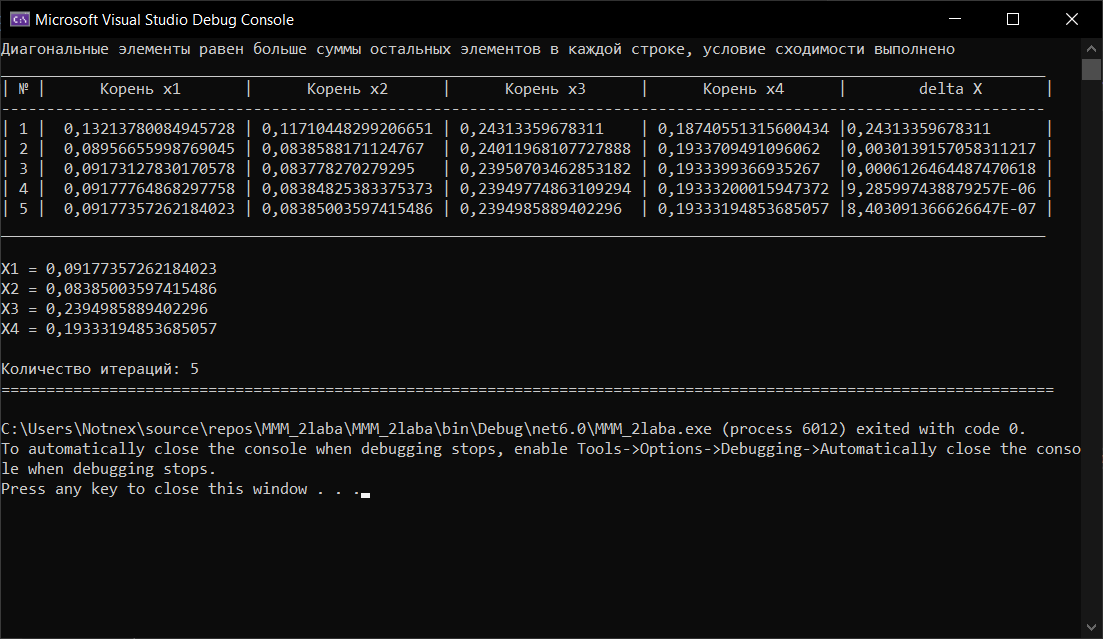
4,238>0,329+0,256+0,425

2,964>0,249+0,351+0,127

2,897>0,365+0,217+0,168

3,127>0,178+0,294+0,432

следовательно, условие сходимости выполнено, и систему можно решать методом Зейделя.



**Код:**

class Program

{

static void Main(string[] args)

{

int n = 4;

double[,] m = new double[n, n];

double[] f = new double[n];

double[] x = new double[n]; //нулевые приближения

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

{

x[i] = 0;

}

m[0, 0] = 4.238;

m[0, 1] = 0.329;

m[0, 2] = 0.256;

m[0, 3] = 0.425;

f[0] = 0.56;

m[1, 0] = 0.249;

m[1, 1] = 2.964;

m[1, 2] = 0.351;

m[1, 3] = 0.127;

f[1] = 0.38;

m[2, 0] = 0.365;

m[2, 1] = 0.217;

m[2, 2] = 2.897;

m[2, 3] = 0.168;

f[2] = 0.778;

m[3, 0] = 0.178;

m[3, 1] = 0.294;

m[3, 2] = 0.432;

m[3, 3] = 3.127;

f[3] = 0.749;

GaussZeidel test = new GaussZeidel(m, f, 500, n, x);

bool IsDiagonal = test.DiagonallyDominant();

test.algoritm();

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

{

Console.WriteLine("X" + (j + 1) + " = " + test.roots[j]);

}

Console.WriteLine("\nКоличество итераций: " + test.k + "\n=====================================================================================================================");

}

}

public class GaussZeidel

{

public static double epsilon = 1e-6; //точность вычисления

public int n, k, N; //N -допустимое число итераций, n - размерность квадратной матрицы коэффицентов, k-количество итераций

public double s, Xi, diff = 1; //s - сумма, величина погрешности

public double[,] matrix; //матрица коэффицентов

public double[] value; //матрица значений

public double[] roots; //матрица корней

public bool diagonal;

public GaussZeidel(double[,] matrix, double[] value, int N, int n, double[] roots)

{

this.matrix = matrix;

this.N = N;

this.value = value;

this.n = n;

this.roots = roots;

}

public bool DiagonallyDominant()

{

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

{

double sum = 0;

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

{

if (i != j)

{

sum += Math.Abs(matrix[i, j]);

}

}

if (Math.Abs(matrix[i, i]) >= sum)

{

diagonal = true;

Console.WriteLine("Диагональный элемент равен " + matrix[i, i]+" сумма остальных элементов строки "+sum);

break;

}

else

{

diagonal = false;

}

}

return diagonal;

}

double max = 0;

double max1 = 0;

public void algoritm()

{

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| № | Корень x1 | Корень x2 | Корень x3 | Корень x4 | delta X |");

Console.WriteLine("--------------------------------------------------------------------------------------------------------------------");

k = 0;

while ((k <= N) && (diff >= epsilon))

{

Console.Write("| {0} | ", k + 1);

k = k + 1;

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

{

s = 0;

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

{

if (i != j)

{

s += matrix[i, j] \* roots[j];

}

}

Xi = (value[i] - s) / matrix[i, i];

diff = Math.Abs(Xi - roots[i]);

roots[i] = Xi;

Console.Write(" {0, -19} |", Xi);

if (Xi > max)

{

max = Xi;

}

}

Console.WriteLine("{0, -21} |", Math.Abs(max-max1));

max1 = max;

max = 0;

}

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

}

}

**Вывод:** Метод Гаусса прост, но недостатком этого метода является невозможность сформулировать условия совместности и определенности системы в зависимости от значений коэффицентов и свободных членов. Метод Зейделя так же прост, но более медленный , т.к. является итерационным.