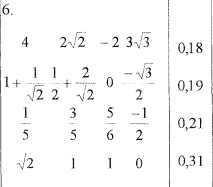
Постановка задачи:

Решить систему линейных алгебраических уравнений:   
  
при условии, что ,,и вычисляются с 2, 4, 6, 10   
знаками после запятой. Полученные результаты подставлять в   
систему уравнений, в которой все корни вычисляются с ма­шинной точностью.  
1. Путём обычного метода Гаусса  
2. Путём модифицированного метода Гаусса с выбором главного элемента.

Текст программы 1:  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace chislennie\_1

{

class Program

{

public static double[,] Multiply(double[,] matrix1, double[,] matrix2)

{

var matrix1Rows = matrix1.GetLength(0);

var matrix1Cols = matrix1.GetLength(1);

var matrix2Rows = matrix2.GetLength(0);

var matrix2Cols = matrix2.GetLength(1);

double[,] product = new double[matrix1Rows, matrix2Cols];

for (int matrix1\_row = 0; matrix1\_row < matrix1Rows; matrix1\_row++)

{

for (int matrix2\_col = 0; matrix2\_col < matrix2Cols; matrix2\_col++)

{

for (int matrix1\_col = 0; matrix1\_col < matrix1Cols; matrix1\_col++)

{

product[matrix1\_row, matrix2\_col] += matrix1[matrix1\_row, matrix1\_col] \* matrix2[matrix1\_col, matrix2\_col];

}

}

}

return product;

}

public static double[,] SubMatrix(double[,] matrix1, double[,] matrix2)

{

var matrixRows = matrix1.GetLength(0);

var matrixCols = matrix1.GetLength(1);

double[,] result = new double[matrixRows, matrixCols];

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

{

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

{

result[i, j] = matrix1[i, j] - matrix2[i, j];

}

}

return result;

}

static void PrintMatrix(double[,] matrixA, double[,] matrixB, int round)

{

Console.WriteLine("Матрица A :");

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

{

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

{

double elemA = Math.Round(matrixA[i, j], round);

if (elemA >= 0)

{ Console.Write(" "); }

Console.Write("{0:f" + round + "}", elemA);

Console.Write(" ");

}

Console.Write(" | ");

Console.Write("{0:f" + round + "}", matrixB[i, 0]);

Console.WriteLine();

}

}

static void Main(string[] args)

{

const int n = 4;

for (int j = 2; j <= 10; j = j + 2)

{

int round = j;

Console.WriteLine($"Округление с точностью до {round} знаков :");

double[,] matrixA = new double[n, n]

{

{4, Math.Round(2 \* Math.Sqrt(2) / 2 ,round), -2 , Math.Round(3 \* Math.Sqrt(3), round)},

{Math.Round(1D / 5D, round), Math.Round(3D / 5D, round), Math.Round(5D / 6D , round), Math.Round(-1D / 2D, round) },

{Math.Round(Math.Sqrt(2), round), 1, 1, 0 },

{Math.Round(1 + 1D / Math.Sqrt(2), round), Math.Round(1D / 2 + 2D / Math.Sqrt(2), round), 0, Math.Round(-Math.Sqrt(3) / 2, round)},

};

double[,] matrixACopy = new double[n, n]

{

{4, 2 \* Math.Sqrt(2) / 2, -2 , 3 \* Math.Sqrt(3)},

{1D / 5D, 3D / 5D, 5D / 6D, -1D / 2D},

{Math.Sqrt(2), 1, 1, 0 },

{1 + 1D / Math.Sqrt(2), 1D / 2 + 2D / Math.Sqrt(2), 0, -Math.Sqrt(3) / 2},

};

double[,] matrixB = new double[n, 1]

{

{Math.Round(0.18,round) },

{Math.Round(0.19,round) },

{Math.Round(0.21 ,round)},

{Math.Round(0.31 ,round)}

};

double[,] matrixBCopy = new double[n, 1]

{

{0.18},

{0.19},

{0.21},

{0.31}

};

PrintMatrix(matrixA, matrixB, round);

double[,] X = new double[,] { { 0 }, { 0 }, { 0 }, { 0 } };

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

{

double mainRowElem = Math.Round(matrixA[i, i], round);

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

{

matrixA[i, q] = Math.Round(Math.Round(matrixA[i, q], round) / Math.Round(mainRowElem, round), round);

}

matrixB[i, 0] = Math.Round(Math.Round(matrixB[i, 0], round) / Math.Round(mainRowElem, round), round);

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

{

double v = Math.Round(matrixA[p, i], round);

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

{

matrixA[p, q] = Math.Round(Math.Round(matrixA[p, q], round) - Math.Round(matrixA[i, q], round) \* Math.Round(v, round), round);

}

matrixB[p, 0] = Math.Round(Math.Round(matrixB[p, 0], round) - Math.Round(matrixB[i, 0], round) \* Math.Round(v, round), round);

}

}

X[3, 0] = Math.Round(matrixB[3, 0], round);

X[2, 0] = Math.Round(matrixB[2, 0] - Math.Round(matrixA[2, 3], round) \* Math.Round(X[3, 0], round), round);

X[1, 0] = Math.Round(matrixB[1, 0] - Math.Round(matrixA[1, 2], round) \* Math.Round(X[2, 0], round) - Math.Round(matrixA[1, 3], round) \* Math.Round(X[3, 0], round), round);

X[0, 0] = Math.Round(matrixB[0, 0] - Math.Round(matrixA[0, 1], round) \* Math.Round(X[1, 0], round) - Math.Round(matrixA[0, 2], round) \* Math.Round(X[2, 0], round) - Math.Round(matrixA[0, 3], round) \* Math.Round(X[3, 0], round), round);

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

{

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

Console.Write("{0:f19}", X[i, 0]);

Console.WriteLine();

}

double[,] AX = Multiply(matrixACopy, X);

double[,] nevyazka = SubMatrix(AX, matrixBCopy);

Console.WriteLine("\nСтолбец невязки (AX - B):");

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

{

Console.Write("{0:f19}", nevyazka[i, 0]);

Console.WriteLine();

}

Console.WriteLine("\n");

}

}

}

}

Текст программы 2:  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace chislennie\_12

{

public class Program

{

public static double[,] Multiply(double[,] matrix1, double[,] matrix2)

{

var matrix1Rows = matrix1.GetLength(0);

var matrix1Cols = matrix1.GetLength(1);

var matrix2Rows = matrix2.GetLength(0);

var matrix2Cols = matrix2.GetLength(1);

double[,] product = new double[matrix1Rows, matrix2Cols];

for (int matrix1\_row = 0; matrix1\_row < matrix1Rows; matrix1\_row++)

{

for (int matrix2\_col = 0; matrix2\_col < matrix2Cols; matrix2\_col++)

{

for (int matrix1\_col = 0; matrix1\_col < matrix1Cols; matrix1\_col++)

{

product[matrix1\_row, matrix2\_col] +=

matrix1[matrix1\_row, matrix1\_col] \* matrix2[matrix1\_col, matrix2\_col];

}

}

}

return product;

}

public static double[,] SubMatrix(double[,] matrix1, double[,] matrix2)

{

var matrixRows = matrix1.GetLength(0);

var matrixCols = matrix1.GetLength(1);

double[,] result = new double[matrixRows, matrixCols];

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

{

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

{

result[i, j] = matrix1[i, j] - matrix2[i, j];

}

}

return result;

}

static void PrintMatrix(double[,] matrixA, double[,] matrixB, int round)

{

Console.WriteLine("Матрица A :");

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

{

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

{

double elemA = Math.Round(matrixA[i, j], round);

if (elemA >= 0)

{ Console.Write(" "); }

Console.Write("{0:f" + round + "}", elemA);

Console.Write(" ");

}

Console.Write(" | ");

Console.Write("{0:f" + round + "}", matrixB[i, 0]);

Console.WriteLine();

}

}

static void Main(string[] args)

{

const int n = 4;

for (int j = 2; j <= 10; j = j + 2)

{

int round = j;

Console.WriteLine($"Округление с точностью до {round} знаков :");

double[,] matrixA = new double[n, n]

{

{4, Math.Round(2 \* Math.Sqrt(2) / 2 ,round), -2 , Math.Round(3 \* Math.Sqrt(3), round)},

{Math.Round(1D / 5D, round), Math.Round(3D / 5D, round), Math.Round(5D / 6D , round), Math.Round(-1D / 2D, round) },

{Math.Round(Math.Sqrt(2), round), 1, 1, 0 },

{Math.Round(1 + 1D / Math.Sqrt(2), round), Math.Round(1D / 2 + 2D / Math.Sqrt(2), round), 0, Math.Round(-Math.Sqrt(3) / 2, round)},

};

double[,] matrixACopy = new double[n, n]

{

{4, 2 \* Math.Sqrt(2) / 2, -2 , 3 \* Math.Sqrt(3)},

{1D / 5D, 3D / 5D, 5D / 6D, -1D / 2D},

{Math.Sqrt(2), 1, 1, 0 },

{1 + 1D / Math.Sqrt(2), 1D / 2 + 2D / Math.Sqrt(2), 0, -Math.Sqrt(3) / 2},

};

double[,] matrixB = new double[n, 1]

{

{Math.Round(0.18,round) },

{Math.Round(0.19,round) },

{Math.Round(0.21 ,round)},

{Math.Round(0.31 ,round)}

};

double[,] matrixBCopy = new double[n, 1]

{

{0.18},

{0.19},

{0.21},

{0.31}

};

PrintMatrix(matrixA, matrixB, round);

int[] vectorZ = new int[n] { 0, 1, 2, 3 };

double maxElem = 0;

double tempElem = 0;

int maxElemRowIndex = 0;

int maxElemColIndex = 0;

double[,] X = new double[,] { { 0 }, { 0 }, { 0 }, { 0 } };

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

{

maxElem = 0;

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

{

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

{

double elem = Math.Round(matrixA[p, q], round);

if (Math.Abs(elem) >= maxElem)

{

maxElem = Math.Round(elem, round);

maxElemRowIndex = p;

maxElemColIndex = q;

}

}

}

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

{

double c = Math.Round(matrixA[i, q], round);

matrixA[i, q] = Math.Round(matrixA[maxElemRowIndex, q], round);

matrixA[maxElemRowIndex, q] = Math.Round(c, round);

}

tempElem = Math.Round(matrixB[i, 0], round);

matrixB[i, 0] = Math.Round(matrixB[maxElemRowIndex, 0], round);

matrixB[maxElemRowIndex, 0] = Math.Round(tempElem, round);

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

{

double c = Math.Round(matrixA[p, i], round);

matrixA[p, i] = Math.Round(matrixA[p, maxElemColIndex], round);

matrixA[p, maxElemColIndex] = Math.Round(c, round);

}

int u = vectorZ[i];

vectorZ[i] = vectorZ[maxElemColIndex];

vectorZ[maxElemColIndex] = u;

double mainRowElem = Math.Round(matrixA[i, i], round);

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

{

matrixA[i, q] = Math.Round(Math.Round(matrixA[i, q], round) / Math.Round(mainRowElem, round), round);

}

matrixB[i, 0] = Math.Round(Math.Round(matrixB[i, 0], round) / Math.Round(mainRowElem, round), round);

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

{

double v = Math.Round(matrixA[p, i], round);

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

{

matrixA[p, q] = Math.Round(Math.Round(matrixA[p, q], round) - Math.Round(matrixA[i, q], round) \* Math.Round(v, round), round);

}

matrixB[p, 0] = Math.Round(Math.Round(matrixB[p, 0], round) - Math.Round(matrixB[i, 0], round) \* Math.Round(v, round), round);

}

}

X[vectorZ[3], 0] = Math.Round(matrixB[3, 0], round);

X[vectorZ[2], 0] = Math.Round(matrixB[2, 0] - Math.Round(matrixA[2, 3], round) \* Math.Round(X[vectorZ[3], 0], round), round);

X[vectorZ[1], 0] = Math.Round(matrixB[1, 0] - Math.Round(matrixA[1, 2], round) \* Math.Round(X[vectorZ[2], 0], round) - Math.Round(matrixA[1, 3], round) \* Math.Round(X[vectorZ[3], 0], round), round);

X[vectorZ[0], 0] = Math.Round(matrixB[0, 0] - Math.Round(matrixA[0, 1], round) \* Math.Round(X[vectorZ[1], 0], round) - Math.Round(matrixA[0, 2], round) \* Math.Round(X[vectorZ[2], 0], round) - Math.Round(matrixA[0, 3], round) \* Math.Round(X[vectorZ[3], 0], round), round);

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

{

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

Console.Write("{0:f19}", Math.Round(X[i, 0], round));

Console.WriteLine();

}

double[,] AX = Multiply(matrixACopy, X);

double[,] nevyazka = SubMatrix(AX, matrixBCopy);

Console.WriteLine("\nСтолбец невязки (AX - B):");

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

{

Console.WriteLine("{0:f19}", nevyazka[i, 0]);

}

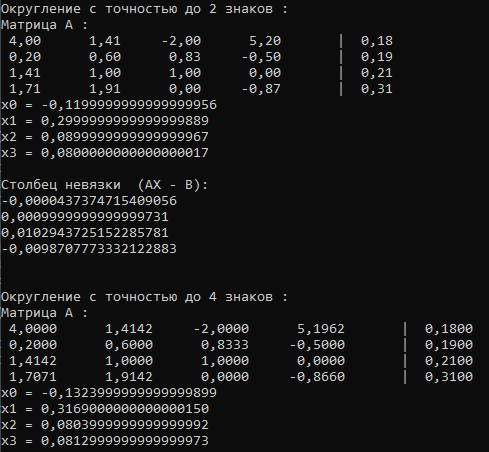
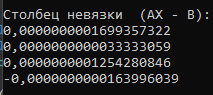
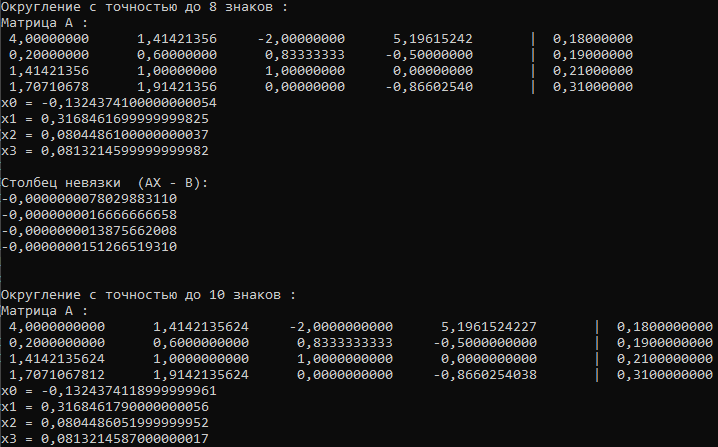
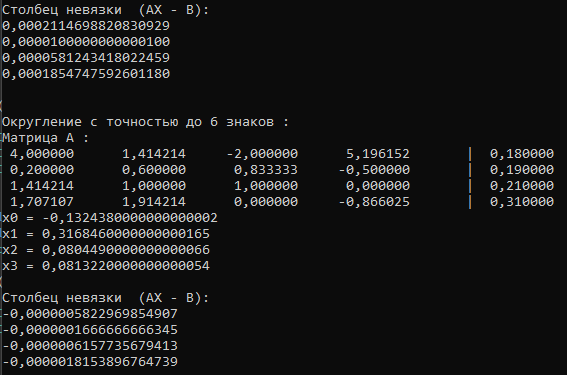
Console.WriteLine("\n");

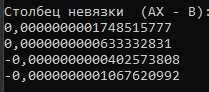
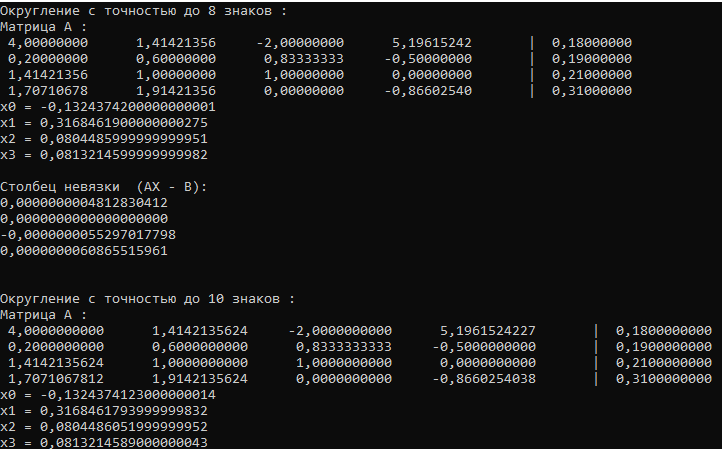
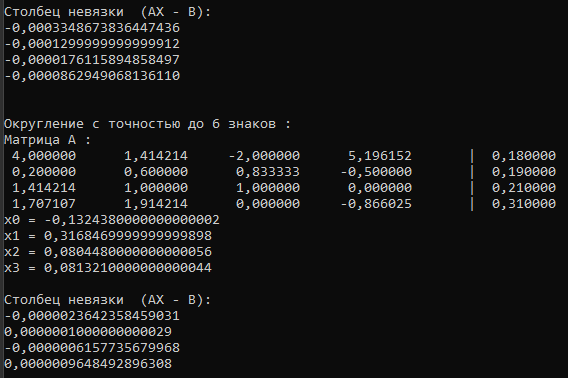
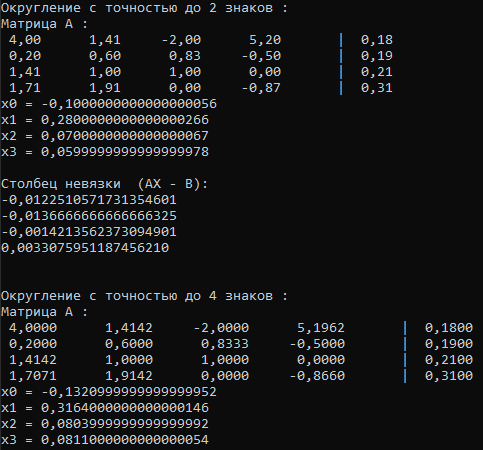
}

}

}

}

Результаты тестирования программы 1:   


Резельтаты работы программы 2: 

Выводы:

Модифицированный способ Гаусса показывает большую точность результатов с низким значением округления, но когда значение округления увеличивается , то два два способа начинают показывать почти идентичные результаты.