БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и информатики

Никончик Даниил Викторович

ОТЧЕТ ПО МЕТОДАМ ВЫЧИСЛЕНИЙ

студента 2 курса 13 группы

Лабораторная работа №1

Преподаватель
Бондарь И.В.

Минск 2021

Постановка задачи.

Задание 3. Метод Гаусса с выбором ГЭ по всей матрице

1. Написать программу, которая решает СЛАУ A x=b и вычисляет определитель матрицы A методом Гаусса с выбором главного элемента по всей матрице. Применить программу к следующим ниже входным данным и вывести результат.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \\ 1 & 3 & 9 & 27 & 81 & 243 & 729 & 2187 & 6561 \\ 1 & 4 & 16 & 64 & 256 & 1024 & 4096 & 16384 & 65536 \\ 1 & 5 & 25 & 125 & 625 & 3125 & 15625 & 78125 & 390625 \\ 1 & 6 & 36 & 216 & 1296 & 7776 & 46656 & 279936 & 1679616 \\ 1 & 7 & 49 & 343 & 2401 & 16807 & 117649 & 823543 & 5764801 \\ 1 & 8 & 64 & 512 & 4096 & 32768 & 262144 & 2097152 & 16777216 \\ 1 & 9 & 81 & 729 & 6561 & 59049 & 531441 & 4782969 & 43046721 \end{pmatrix} \quad b = \begin{pmatrix} 9 \\ 511 \\ 9841 \\ 87381 \\ 488281 \\ 2015539 \\ 6725601 \\ 19173961 \\ 48427561 \end{pmatrix}$$

$$A = \begin{pmatrix} -3 & -1 & -1 & 1 & 4 & 2 & 3 & -4 & -1 \\ -3 & -1 & 4 & 1 & 0 & 3 & -5 & 5 & 5 \\ -6 & -2 & 3 & 5 & 3 & -3 & -4 & -3 & 1 \\ -12 & -4 & 6 & 7 & 2 & 0 & 1 & -2 & -1 \\ -24 & -8 & 12 & 14 & 9 & 4 & 4 & -1 & 5 \\ -48 & -16 & 24 & 28 & 18 & 6 & -2 & 1 & -2 \\ -96 & -32 & 48 & 56 & 36 & 12 & -3 & 2 & 4 \\ -192 & -64 & 96 & 112 & 72 & 24 & -6 & -2 & 5 \\ 0 & -1 & 3 & 0 & 2 & 2 & -5 & -3 & 4 \end{pmatrix} \quad b = \begin{pmatrix} 8 \\ 79 \\ -27 \\ 18 \\ 186 \\ 206 \\ 491 \\ 907 \\ 6 \end{pmatrix}$$

$$A = \begin{pmatrix} 2 & 0 & -5 & -3 & 1 & 1 & 3 & 2 & 2 \\ 2 & 0 & -5 & -5 & 2 & -1 & 4 & -2 & -3 \\ 4 & 0 & -10 & -1 & 5 & -3 & 1 & -5 & -5 \\ 8 & 0 & -20 & -9 & 5 & 1 & 1 & -5 & 4 \\ 16 & 0 & -40 & -18 & 13 & 1 & -1 & 0 & -3 \\ 32 & 0 & -80 & -36 & 26 & -1 & 0 & -1 & -5 \\ 64 & 0 & -160 & -72 & 52 & -2 & 8 & 0 & -2 \\ 128 & 0 & -320 & -144 & 104 & -4 & 16 & -11 & 5 \\ -1 & 1 & 1 & 3 & 0 & -3 & 0 & -5 & 3 \end{pmatrix} \quad b = \begin{pmatrix} 41 \\ -44 \\ -101 \\ -54 \\ -139 \\ -281 \\ -418 \\ -843 \\ -15 \end{pmatrix}$$

2. Найти точное решение указанных систем с использованием библиотеки SymPy или аналогичного программного обеспечения и сравнить результаты.

3. Вычислить число обусловленности в максимум-норме матрицы A из второго тестового задания. Что это означает на практике? Путем решения нескольких СЛАУ с возмущенным вектором b подтвердите связь между числом обусловленности и относительными погрешностями начальных данных и решения.

4. Проведите экспериментальное исследование скорости решения СЛАУ в зависимости от размерности системы, используя для тестов матрицу A и вектор b со случайными числами. Постройте график зависимости времени работы от размерности. Систему какой размерности ваша программа на вашем компьютере может решить за одну минуту?

Код и комментарии.

Прямой ход метода Гаусса

```
private void TDCOTGM() // The direct course of the Gauss method
{
    for (int i = 0; i < size; i++)
    {
        FTMEITM(i);
        if (Math.Abs(Matrix[i,i]) == 0.00000001)
        {
            throw new GaussMethodSolutionNotFound("Determinant are singular");
        }

        for (int j = i + 1; j < size; j++)
        {
            double coeff = Matrix[j,i]/Matrix[i, i];
            for (int k = i + 1; k < size; k++)
            {
                Matrix[j, k] -= Matrix[i, k] * coeff;
            }
            VectorB[j] -= VectorB[i] * coeff;
        }
    }
}
```

Обратный ход метода гаусса.

```
private double[] RCOTGM() // Reverse course of the Gauss method
{
    double[] result = new double[size];

    for (int i = size - 1; i >= 0; i--)
    {
        for (int j = i + 1; j < size; j++)
        {
            VectorB[i] -= Matrix[i, j] * result[j];
        }
        result[i] = VectorB[i] / Matrix[i, i];
    }
    return result;
}
```

Подсчет определителя

```
public double CountingDeterminant() // Counting the determinant
{
    double result = 1;

    for (int i = 0; i < size; i++)
    {
        result *= Matrix[i, i];
    }

    return result;
}
```

Поиск максимального элемента в матрице.

```
private void FTMEITM(int dI) // Finding the maximum element in the matrix
{
    int maxI = 0;
    int maxJ = 0;
    double maxAbs = 0;

    for (int i = dI; i < size; i++)
    {
        for (int j = dI; j < size; j++)
        {
            if (maxAbs < Math.Abs(Matrix[i, j]))
            {
                maxAbs = Math.Abs(Matrix[i, j]);
                maxI = i;
                maxJ = j;
            }
        }
    }

    RRAC(dI, maxI, maxJ);
}
```

Перестановка строк и столбцов.

```
private void RRAC(int i, int j, int dI) // Rearranging rowns and columns
{
    for (int k = 0; k < size; k++)
    {
        double tmpI = Matrix[k, dI];
        Matrix[k, dI] = Matrix[k, j];
        Matrix[k, j] = tmpI;
    }

    for (int k = 0; k < size; k++)
    {
        double tmpJ = Matrix[dI, k];
        Matrix[dI, k] = Matrix[i, k];
        Matrix[i, k] = tmpJ;
    }

    double tmpB = VectorB[dI];
    VectorB[dI] = VectorB[i];
    VectorB[i] = tmpB;

    int tmpX = transpositionX[dI];
    transpositionX[dI] = transpositionX[j];
    transpositionX[j] = tmpX;
}
```

Возврат вектора X и определителя.

```
public double[] ReturnVectorDeterminant() // Return of the vector X and determinant
{
    InitTransposition();
    TDCOTGM();
    double[] tmpVector = RCOTGM();

    double[] result = new double[size];
    for (int i = 0; i < size; i++)
    {
        result[i] = tmpVector[transpositionX[i]];
    }

    return result;
}
```

Решение брались с сайта matworld.ru.

Пример 1

```
**** Matrix A ****
1       1       1       1       1       1       1       1       1
1       2       4       8       16      32      64      128     256
1       3       9       27      81      243     729     2187    6561
1       4       16      64      256     1024    4096    16384   65536
1       5       25      125     625     3125    15625   78125   390625
1       6       36      216     1296    7776    46656   279936  1679616
1       7       49      343     2401    16807   117649  823543  5764801
1       8       64      512     4096    32768   262144  2097152 16777216
1       9       81      729     6561    59049   531441  4782969 43046721

**** Vector b ****
9 511 9841 87381 488281 2015539 6725601 19173961 48427561

**** Result of solution ****
1 1 1 1 1 1 1 1 1

**** Determinant ****
-5056584744964090
```

Подставив нижние выражения в верхние, получим **решение**.

$$x_1=1$$
$$x_2=1$$
$$x_3=1$$
$$x_4=1$$
$$x_5=1$$
$$x_6=1$$
$$x_7=1$$
$$x_8=1$$
$$x_9=1$$

Пример 2

```
**** Matrix A ****
-3      -1      -1      1       4       2       3       -4      -1
-3      -1      4       1       0       3       -5      5       5
-6      -2      3       5       3       -3      -4      -3      1
-12     -4      6       7       2       0       1       -2      -1
-24     -8      12      14      9       4       4       -1      5
-48     -16     24      28      18      6       -2      1       -2
-96     -32     48      56      36      12      -3      2       4
-192    -64     96      112     72      24      -6      -2      5
0       -1      3       0       2       2       -5      -3      4

**** Vector b ****
8 79 -27 18 186 206 491 907 6

**** Result of solution ****
1 2 3 4 5 6 7 8 9

**** Determinant ****
16200
```

Подставив нижние выражения в верхние, получим **решение**.

$$x_1=1$$
$$x_2=2$$
$$x_3=3$$
$$x_4=4$$
$$x_5=5$$
$$x_6=6$$
$$x_7=7$$
$$x_8=8$$
$$x_9=9$$

Пример 3

```
**** Matrix A ****
2       0       -5      -3      1       1       3       2       2
2       0       -5      -5      2       -1      4       -2      -3
4       0       -10     -1      5       -3      1       -5      -5
8       0       -20     -9      5       1       1       -5      4
16      0       -40     -18     13      1       -1      0       -3
32      0       -80     -36     26      -1      0       -1      -5
64      0       -160    -72     52      -2      8       0       -2
128     0       -320    -144    104     -4      16      -11     5
-1      1       1       3       0       -3      0       -5      3

**** Vector b ****
41 -44 -101 -54 -139 -281 -418 -843 -15
Solution can't be found: Determinant are singular
```

Подставив нижние выражения в верхние, получим **решение**.

$$x_1 = -\frac{13}{2} + \frac{5}{2} \cdot x_3$$
$$x_2 = -\frac{5}{2} + \frac{3}{2} \cdot x_3$$
$$x_4 = 4$$
$$x_5 = 5$$
$$x_6 = 6$$
$$x_7 = 7$$
$$x_8 = 8$$
$$x_9 = 9$$

где x_3 – произвольное действительное число.

Сделаем следующие обозачения:

$$x_3 = \lambda_1.$$

Тогда

$$x_1 = -\frac{13}{2} + \frac{5}{2} \cdot \lambda_1$$
$$x_2 = -\frac{5}{2} + \frac{3}{2} \cdot \lambda_1$$
$$x_4 = 4$$
$$x_5 = 5$$
$$x_6 = 6$$
$$x_7 = 7$$
$$x_8 = 8$$
$$x_9 = 9$$
$$x_3 = \lambda_1.$$

где λ_1 – произвольное действительное число.

Задание 3

Найдем оценку числа обусловленности используя формулу:

$$cond(A) \geq \frac{\|\Delta x\| / \|x\|}{\|\Delta b\| / \|b\|}$$

Где **Δb** равно 0.01

```
**** Result of solution № 1****
1 2 3 4 5 6 7 8 9

**** Result of solution № 2****
0,938 2,098 3,068 3,878 5,041 5,945 7,013 8,001 8,998

**** Condition number ****
1980,484
```

Получаем ответы.

Решение СЛАУ с возмущенным вектором b.
(От каждого элемента вектора b по очереди отнимал 0.01: $b_i = b_i - 0.01$)

```
**** Result of solution with variable vector № 1 ****
1,062 1,902 2,932 4,123 4,959 6,055 6,987 7,999 9,002

**** Result of solution with variable vector № 2 ****
1,06 1,914 2,936 4,123 4,959 6,055 6,987 7,999 9,002

**** Result of solution with variable vector № 3 ****
1,065 1,908 2,934 4,129 4,959 6,055 6,987 7,999 9,002

**** Result of solution with variable vector № 4 ****
1,065 1,891 2,931 4,125 4,955 6,055 6,987 7,999 9,002

**** Result of solution with variable vector № 5 ****
1,077 1,896 2,928 4,151 4,953 6,065 6,987 7,999 9,002

**** Result of solution with variable vector № 6 ****
1,178 1,746 2,829 4,343 4,891 6,15 6,967 7,999 9,002

**** Result of solution with variable vector № 7 ****
1,019 1,947 2,977 4,027 4,989 6,01 6,997 8,003 9,002

**** Result of solution with variable vector № 8 ****
0,928 2,115 3,071 3,866 5,044 5,942 7,013 7,999 8,998

**** Result of solution with variable vector № 9 ****
1,003 1,99 3 4 5 6 7 8 9
```

Число обусловленности говорит о том, насколько сильно изменится решение при малом изменение начальных данных.

Сравнение нормы ошибки в решение с нормой ошибки в исходных данных

```
Result of comparison:
Solution error rate > Error rate in the source date
```

Добавил вот такой код

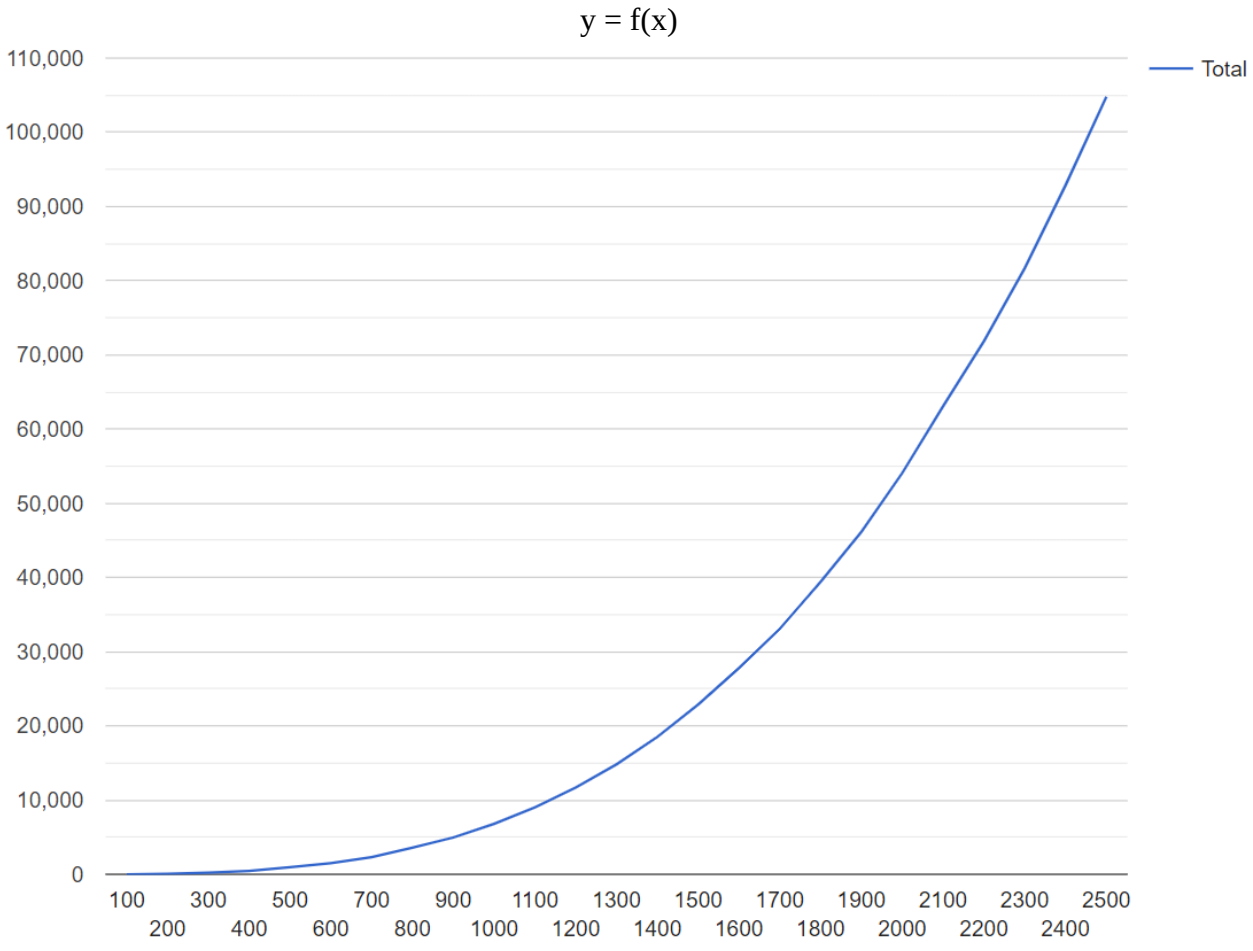
```
if (Norm1 > Norm2)
{
    Console.WriteLine("\nResult of comparison: ");
    Console.WriteLine("Error rate in the source date > Solution error rate \n");
}

if (Norm1 < Norm2)
{
    Console.WriteLine("\nResult of comparison: ");
    Console.WriteLine("Solution error rate > Error rate in the source date\n");
}
```

Задание 4

Зависимость размерности от времени.

Размерность	Миллисекунды
100	7
200	57

300	205
400	454
500	943
600	1512
700	2312
800	3595
900	4963
1000	6794
1100	9019
1200	11693
1300	14821
1400	18523
1500	22904
1600	27787
1700	33085
1800	39436
1900	46156
2000	54061
2100	63074
2200	71807
2300	81636
2400	92827
2500	104796



Как видно из таблицы, моя код решает за минуту систему размерности 2100x2100.

Код программы

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;

namespace ConsoleApp1
{


    class Program
    {
        public static void PrintMatrix(double[,] matrix)
        {
            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(0); j++)
                {
                    Console.Write(matrix[i, j].ToString("0.###\t"));
                }

                Console.WriteLine();
            }
        }

        public static void PrintVector(double[] vector)
        {
            for (int i = 0; i < vector.Length; i++)
            {
                Console.Write(vector[i].ToString("0.### "));
            }

            Console.WriteLine();
        }

        public static double[,] InitMatrix()
        {
            double[,] result;

            using (StreamReader sr = new StreamReader("Matrix.txt"))
            {
                List<string> strLines = new List<string>();
                string line = sr.ReadLine();

                while (line != null)
                {
                    strLines.Add(line);
                    line = sr.ReadLine();
                }

                result = new double[strLines.Count, strLines.Count];
                for (int i = 0; i < strLines.Count; i++)
                {
                    var tmpArr = strLines[i].Split(' ').
                        Select(x => double.Parse(x)).ToArray();
                    for (int j = 0; j < strLines.Count; j++)
                    {
                        result[i, j] = tmpArr[j];
                    }
                }
            }

            return result;
```

```
}

public static double[] InitVector()
{
    double[] result;

    using (StreamReader sr = new StreamReader("Vector.txt"))
    {
        result = sr.ReadLine().Split(' ').Select(x => double.Parse(x)).ToArray();
    }

    return result;
}

private static double GetNorma(double[] vector) =>
    vector.Aggregate((x, y) => Math.Abs(x) + Math.Abs(y));

private static double[] VectorDifference(double[] a, double[] b) =>
    new double[a.Length].Select((x, i) => x = a[i] - b[i]).ToArray();

private static void ConditionTest(double[,] matrix, double[] vector)
{
    double[] res1, res2;
    var vector2 = (vector.Clone() as double[]);
    vector2[0] -= 0.01;
    try
    {
        double determ;
        res1 = new GaussMethod(matrix, vector).GetSolution(out determ);
        res2 = new GaussMethod(matrix, vector2).GetSolution(out determ);
    }
    catch (GaussMethodSolutionNotFound e)
    {
        Console.WriteLine(e.Message);
        Console.ReadKey();
        return;

    }

    Console.WriteLine("\n**** Result of solution № 1****");
    PrintVector(res1);

    Console.WriteLine("\n**** Result of solution № 2****");
    PrintVector(res2);

    var diff1 = VectorDifference(vector, vector2);
    var diff2 = VectorDifference(res1, res2);

    var Norm1 = GetNorma(diff1) / GetNorma(vector);
    var Norm2 = GetNorma(diff2) / GetNorma(res1);

    if (Norm1 > Norm2)
    {
        Console.WriteLine("\nResult of comparison: ");
        Console.WriteLine("Error rate in the source date > Solution error rate \n");
    }

    if (Norm1 < Norm2)
    {
        Console.WriteLine("\nResult of comparison: ");
        Console.WriteLine("Solution error rate > Error rate in the source date\n");
    }

    var cond = Norm2 / Norm1;
    Console.WriteLine("\n**** Condition number ****");
    Console.WriteLine(cond.ToString("0.0##"));

    try
    {
```

```
                double determ;
                for (int i = 0; i < vector.Length; i++)
                {
                    var tmpVector = (vector.Clone() as double[]);
                    tmpVector[i] -= -0.01;
                    var tmpRes = new GaussMethod(matrix, tmpVector).GetSolution(out determ);

                    Console.WriteLine($"\n**** Result of solution with variable vector № {i +
1} ****");
                    PrintVector(tmpRes);
                }
            }
            catch (GaussMethodSolutionNotFound e)
            {
                Console.WriteLine(e.Message);
                Console.ReadKey();
                return;

            }
        }

        private static double[,] InitRandomMatrix(int size)
        {
            double[,] result = new double[size, size];
            Random rnd = new Random();

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    result[i, j] = rnd.Next(100);
                }
            }

            return result;
        }

        private static double[] InitRandomVector(int size)
        {
            double[] result = new double[size];
            Random rnd = new Random();

            for (int i = 0; i < size; i++)
            {
                result[i] = rnd.Next(100);
            }

            return result;
        }

        private static void TimeTest()
        {
            using (StreamWriter sw = new StreamWriter("Time Statistic.txt"))
            {
                for (int i = 100; i <= 500; i += 100)
                {
                    double[,] matrix = InitRandomMatrix(i);
                    double[] vector = InitRandomVector(i);

                    Stopwatch stopwatch = new Stopwatch();
                    stopwatch.Start();
                    double determ;
                    new GaussMethod(matrix, vector).GetSolution(out determ);
                    stopwatch.Stop();

                    Console.WriteLine($"Time for a matrix with size {i}:
{stopwatch.ElapsedMilliseconds}");
                    sw.WriteLine(stopwatch.ElapsedMilliseconds);
                }
```

```
            }
        }

        static void Main(string[] args)
        {
            double[,] matrix = InitMatrix();
            double[] vector = InitVector();

            Console.WriteLine("**** Matrix A ****");
            PrintMatrix(matrix);
            Console.WriteLine("\n**** Vector b ****");
            PrintVector(vector);

            double determ;
            double[] res;
            try
            {
                res = new GaussMethod(matrix, vector).GetSolution(out determ);

            }
            catch (GaussMethodSolutionNotFound e)
            {
                Console.WriteLine(e.Message);
                Console.ReadKey();
                return;

            }

            Console.WriteLine("\n**** Result of solution ****");
            PrintVector(res);
            Console.WriteLine("\n**** Determinant ****");
            Console.Write(determ.ToString("0.######################\n"));

            ConditionTest(matrix, vector);
            TimeTest();

            Console.ReadKey();
        }
    }
}
```

Код с тестами

```
using System;
using System.Linq;

namespace ConsoleApp1
{
    public class GaussMethodSolutionNotFound : Exception
    {
        public GaussMethodSolutionNotFound(string msg)
            : base($"Solution can't be found: {msg} \n")
        {
        }
    }

    class GaussMethod
    {
        private int[] transpositionX;
        private int size;

        public double[,] Matrix { get; }
        public double[] VectorB { get; }
```

```
private void InitTransposition()
{
    transpositionX = new int[size];
    transpositionX = transpositionX.Select((x, i) => x = i).ToArray();
}

public GaussMethod(double[,] matrix, double[] b)
{
    Matrix = matrix.Clone() as double[,];
    VectorB = b.Clone() as double[];
    size = b.Length;
}

public double[] ReturnVectorDeterminant() // Return of the vector X and determinant
{
    InitTransposition();
    TDCOTGM();
    double[] tmpVector = RCOTGM();

    double[] result = new double[size];
    for (int i = 0; i < size; i++)
    {
        result[i] = tmpVector[transpositionX[i]];
    }

    return result;
}

public double[] GetSolution(out double determinant)
{
    var result = this.ReturnVectorDeterminant();
    determinant = CountingDeterminant();
    return result;
}

public double CountingDeterminant() // Counting the determinant
{
    double result = 1;

    for (int i = 0; i < size; i++)
    {
        result *= Matrix[i, i];
    }

    return result;
}

private void RRAC(int dI, int i, int j) // Rearranging rowns and columns
{
    for (int k = 0; k < size; k++)
    {
        double tmpI = Matrix[k, dI];
        Matrix[k, dI] = Matrix[k, j];
        Matrix[k, j] = tmpI;
    }

    for (int k = 0; k < size; k++)
    {
        double tmpJ = Matrix[dI, k];
        Matrix[dI, k] = Matrix[i, k];
        Matrix[i, k] = tmpJ;
    }

    double tmpB = VectorB[dI];
    VectorB[dI] = VectorB[i];
    VectorB[i] = tmpB;

    int tmpX = transpositionX[dI];
    transpositionX[dI] = transpositionX[j];
```

```
            transpositionX[j] = tmpX;
        }

        private void FTMEITM(int dI) // Finding the maximum element in the matrix
        {
            int maxI = 0;
            int maxJ = 0;
            double maxAbs = 0;

            for (int i = dI; i < size; i++)
            {
                for (int j = dI; j < size; j++)
                {
                    if (maxAbs < Math.Abs(Matrix[i, j]))
                    {
                        maxAbs = Math.Abs(Matrix[i, j]);
                        maxI = i;
                        maxJ = j;
                    }
                }
            }

            RRAC(dI, maxI, maxJ);
        }

        private void TDCOTGM() // The direct course of the Gauss method
        {
            for (int i = 0; i < size; i++)
            {
                FTMEITM(i);
                if (Math.Abs(Matrix[i, i]) == 0.00000001)
                {
                    throw new GaussMethodSolutionNotFound("Determinant are singular");
                }

                for (int j = i + 1; j < size; j++)
                {
                    double coeff = Matrix[j, i] / Matrix[i, i];
                    for (int k = i + 1; k < size; k++)
                    {
                        Matrix[j, k] -= Matrix[i, k] * coeff;
                    }
                    VectorB[j] -= VectorB[i] * coeff;
                }
            }
        }

        private double[] RCOTGM() // Reverse course of the Gauss method
        {
            double[] result = new double[size];

            for (int i = size - 1; i >= 0; i--)
            {
                for (int j = i + 1; j < size; j++)
                {
                    VectorB[i] -= Matrix[i, j] * result[j];
                }
                result[i] = VectorB[i] / Matrix[i, i];
            }
            return result;
        }
    }
}
```