БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Приходько Егор Андреевич

ОТЧЕТ ПО МЕТОДАМ ВЫЧИСЛЕНИЙ

студента 2 курса 13 группы

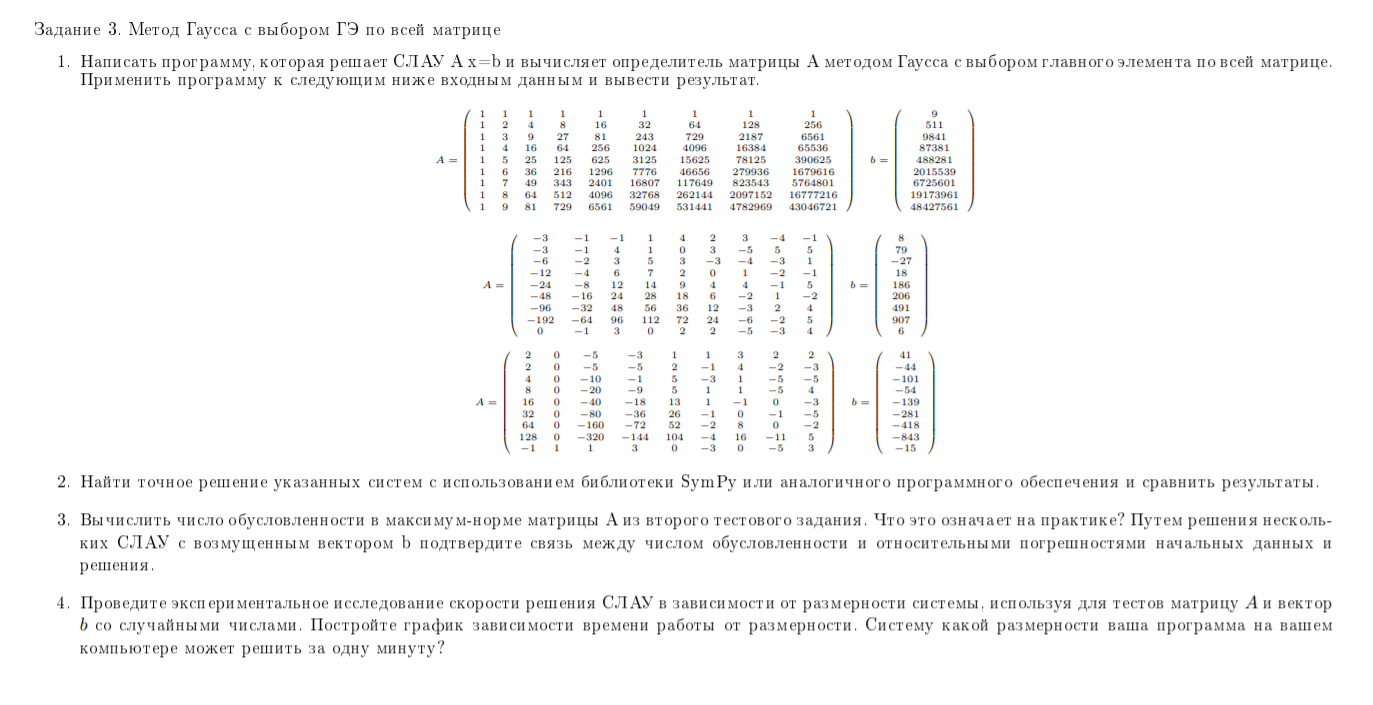
Лабораторная работа №2

Преподаватель

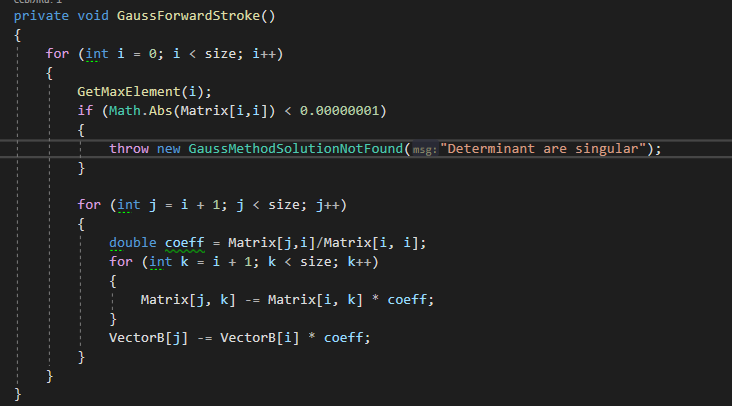
Бондарь И.В.

Минск 2020

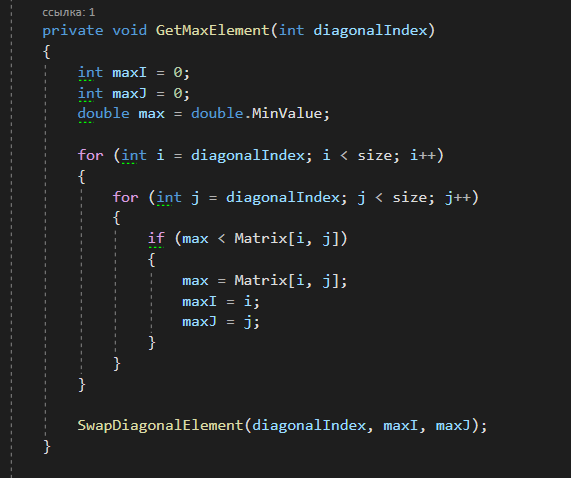
**Постановка задачи.**



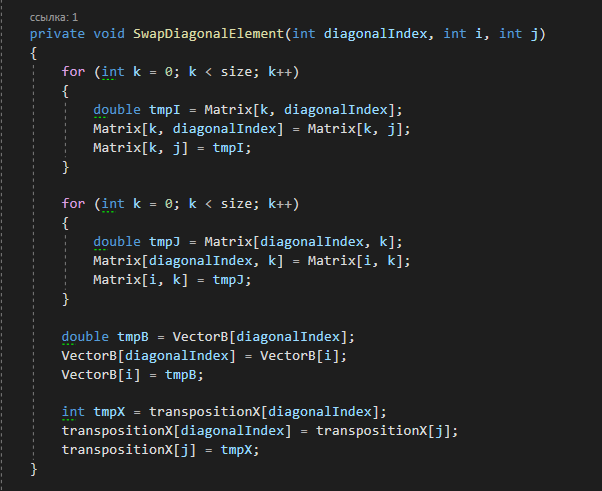
**Код и комментарии.**

Прямой ход метода Гаусса

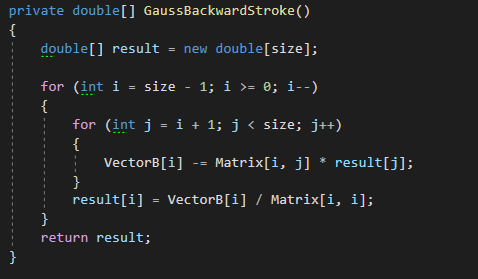
Поиск максимального элемента в матрице.



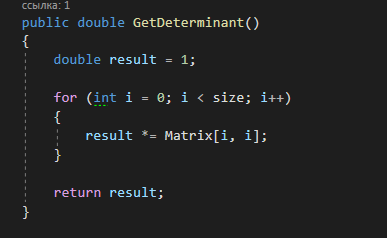
Перестановка строк и столбцов.



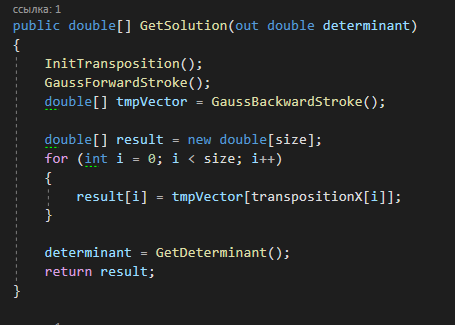
Обратный ход метода гаусса.



Подсчет определителя

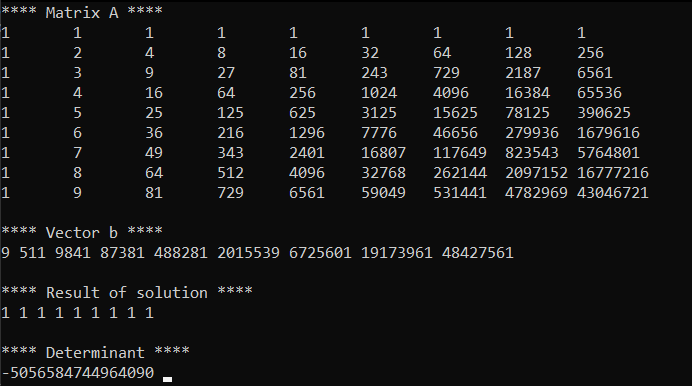
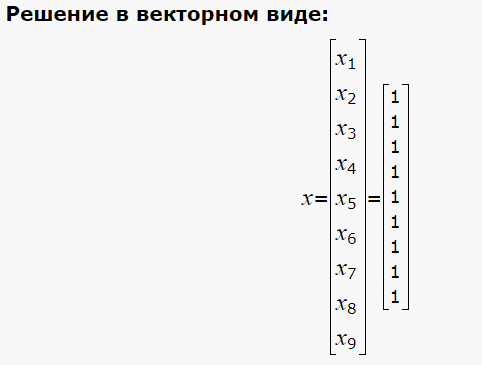


Возврат вектора X и определителя.

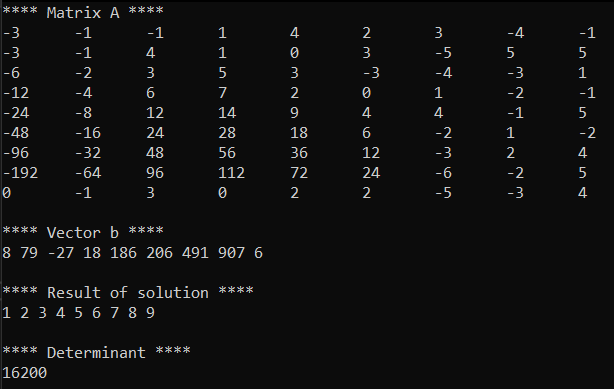
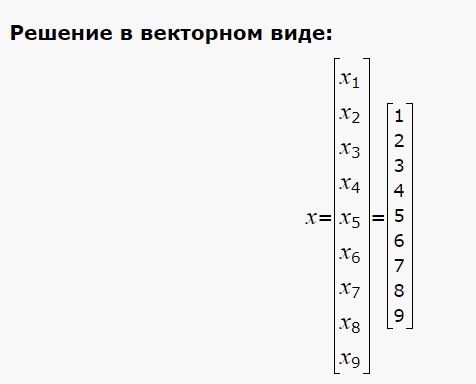


Решение брались с сайта matworld.ru.

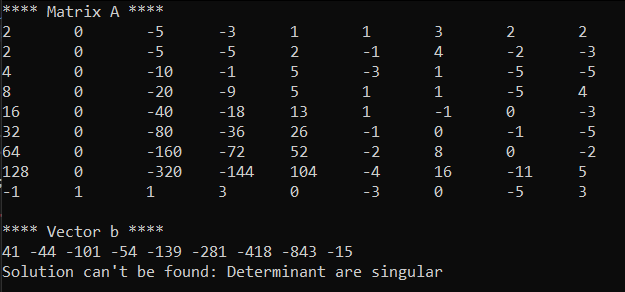
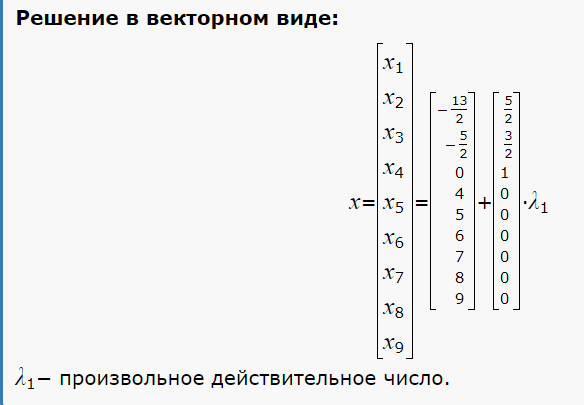
Пример 1

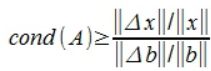
Пример 2

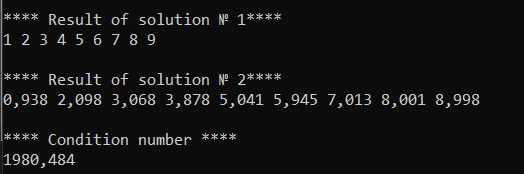
Пример 3

Задание 3  
  
Найдем оценку числа обусловленности используя формулу:



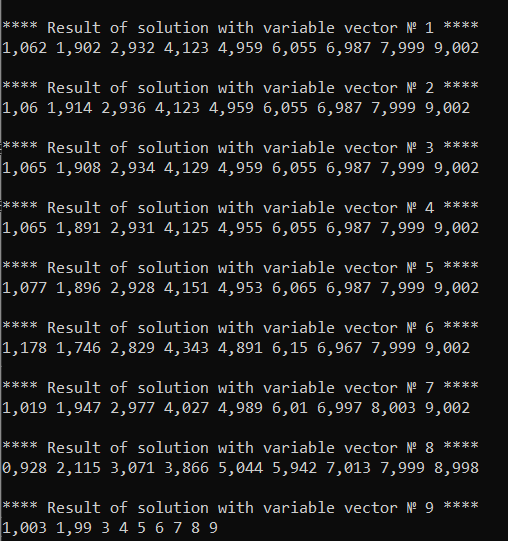
Где **Δb** равно 0.01



Имеем такие результаты.

Решение СЛАУ с возмущенным вектором b.

(От каждого элемента вектора b по очереди отнимал 0.01: bi = bi – 0.01)



Число обусловленности говорит нам о том, насколько сильно изменится решение при малом изменение начальных данных.  
В нашем случае оценка числа получилось довольно большая (почти 2000) и как видим из этого следует что на каждый элемент вектора x погрешность где-то в среднем 6 – 10 %.

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 машина решает за минуту систему размерности 2100x2100.

**Код программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

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[] GetSolution()

{

InitTransposition();

GaussForwardStroke();

double[] tmpVector = GaussBackwardStroke();

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.GetSolution();

determinant = GetDeterminant();

return result;

}

public double GetDeterminant()

{

double result = 1;

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

{

result \*= Matrix[i, i];

}

return result;

}

private void SwapDiagonalElement(int diagonalIndex, int i, int j)

{

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

{

double tmpI = Matrix[k, diagonalIndex];

Matrix[k, diagonalIndex] = Matrix[k, j];

Matrix[k, j] = tmpI;

}

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

{

double tmpJ = Matrix[diagonalIndex, k];

Matrix[diagonalIndex, k] = Matrix[i, k];

Matrix[i, k] = tmpJ;

}

double tmpB = VectorB[diagonalIndex];

VectorB[diagonalIndex] = VectorB[i];

VectorB[i] = tmpB;

int tmpX = transpositionX[diagonalIndex];

transpositionX[diagonalIndex] = transpositionX[j];

transpositionX[j] = tmpX;

}

private void GetMaxElement(int diagonalIndex)

{

int maxI = 0;

int maxJ = 0;

double maxAbs = 0;

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

{

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

{

if (maxAbs < Math.Abs(Matrix[i, j]))

{

maxAbs = Math.Abs(Matrix[i, j]);

maxI = i;

maxJ = j;

}

}

}

SwapDiagonalElement(diagonalIndex, maxI, maxJ);

}

private void GaussForwardStroke()

{

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

{

GetMaxElement(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[] GaussBackwardStroke()

{

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;

}

}

}

**Код с тестами**

﻿using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Dynamic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

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);

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 <= 2500; 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();

}

}

}

**Заключение.**

Можно отметить: число обусловленности очень важно для правильного решение слау, даже важнее чем определитесь.  
Например, в номере 2 определитель довольно большой и можно сказать, что матрица далека от вырожденности, однако все равно мы имеем большую погрешность, потому что число обусловленности равно 1940.