CАНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

**Лабораторная работа по выч.математике №1**

*«Решение СЛАУ методом Гаусса»*

Выполнил: Припадчев Артём

группа 2125

Проверил: Шипилов П.А.

2013 г.

**Задание:** составить программу, обеспечивающую решение системы линейных алгебраических уравнений порядка n (n <=20) методом Гаусса.

**Описание метода**

**Метод Гаусса** основан на приведении матрицы системы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается *x1* из всех последующих уравнений системы. Затем с помощью второго уравнения исключается х2 из третьего и всех последующих уравнений. Этот процесс, называемый *прямым ходом метода Гаусса,* продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным *хп,* т. е. матрица системы будет приведена к треугольному виду (к такому виду приводится лишь невырожденная матрица, в противном случае метод Гаусса неприменим). Когда матрица системы примет треугольный вид закончится прямой ход метода Гаусса.

*Обратный ход метода Гаусса* состоит в последовательном вычислении искомых неизвестных: решая последнее сравнение, находим единственное неизвестное *хп.* Далее, используя это значение, из предыдущего уравнения вычисляем *хп-1* и т. д. Последним найдем *x1* из первого уравнения.

**Расчетные формулы**

*Пусть:*

а – матрица коэффициентов СЛАУ

b – матрица (вектор) свободных членов СЛАУ

i – номер строки

j – номер столбца

m – невязки

x – матрица решений системы

*Тогда:*

i,j = 1..n

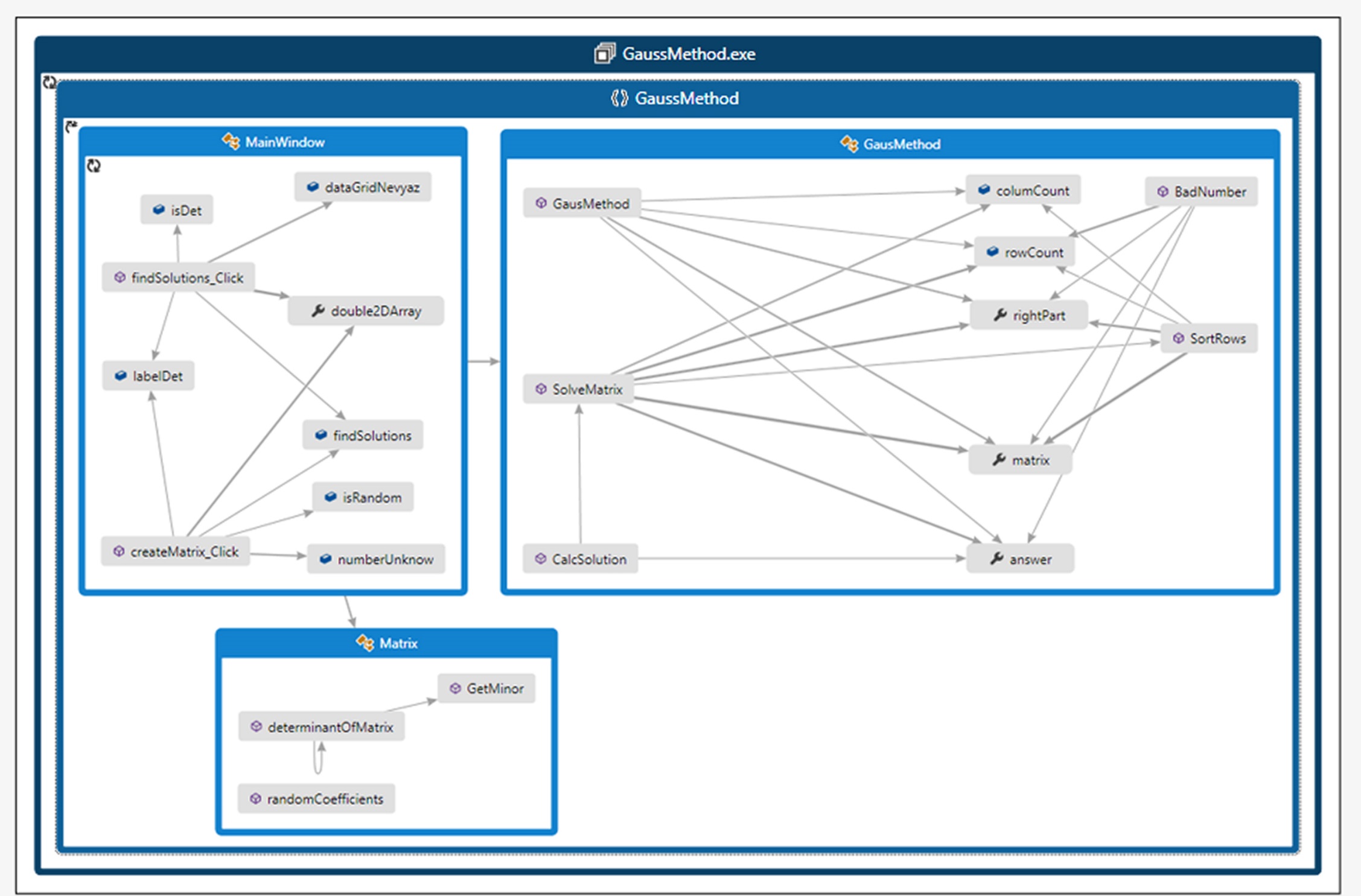
ki = ai(i-1) / a(i-1)(i-1)

a’ij = aij - a(i-1)j\*k

b’i = bi - bi-1\*k

mi = b - ax

**Диаграмма зависимостей**



**Код класса, реализующего решение СЛАУ методом Гаусса**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GaussMethod

{

public class GausMethod

{

protected int rowCount;

protected int columCount;

protected double[,] matrix { get; set; }

protected double[] rightPart { get; set; }

protected double[] answer { get; set; }

//Конструктор класса, принимающий матрицу коэффициентов и столбец свободных членов

public GausMethod(double[,] Matrix)

{

rightPart = new double[Matrix.GetLength(0)];

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

rightPart[i] = Matrix[i, Matrix.GetLength(1) - 1];

answer = new double[Matrix.GetLength(0)];

rowCount = Matrix.GetLength(0);

columCount = Matrix.GetLength(1) - 1; //последний столбец отсекаем, т.к. нам нужны только коэффициенты

this.matrix = new double[Matrix.GetLength(0), Matrix.GetLength(1)];

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

for (int j = 0; j < Matrix.GetLength(1) - 1; j++)

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

}

//Вычисление невязок. Расчетная формула BadNumber[i] = b - Ax

public double[] BadNumber()

{

double[] BadNumber = new double[rowCount];

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

{

double rowCheckSumm = 0;

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

{

rowCheckSumm += matrix[i, j] \* answer[j];

}

BadNumber[i] = rightPart[i] - rowCheckSumm;

}

return BadNumber;

}

//Меняем строки местами, чтобы в вершине стоял максимальный коэффициент, для уменьшения погрешности

private void SortRows(int SortIndex)

{

double MaxElement = matrix[SortIndex, SortIndex];

int MaxElementIndex = SortIndex;

for (int i = SortIndex + 1; i < rowCount; i++)

{

if (matrix[i, SortIndex] > MaxElement)

{

MaxElement = matrix[i, SortIndex];

MaxElementIndex = i;

}

}

//теперь найден максимальный элемент ставим его на верхнее место

if (MaxElementIndex > SortIndex)//если это не первый элемент

{

double Temp;

Temp = rightPart[MaxElementIndex];

rightPart[MaxElementIndex] = rightPart[SortIndex];

rightPart[SortIndex] = Temp;

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

{

Temp = matrix[MaxElementIndex, i];

matrix[MaxElementIndex, i] = matrix[SortIndex, i];

matrix[SortIndex, i] = Temp;

}

}

}

//Метод, возвращающий ответ: null - нет решения, setOfSulutions - множество решений, answer - матрица ответов.

public double[] CalcSolution()

{

double[] setOfSolutions = new double[0];

int result = SolveMatrix();

switch (result)

{

case 1: return null; break;

case 2: return setOfSolutions; break;

case 0: return answer; break;

default: throw new NotImplementedException(); break;

}

}

//Метод, решающий СЛАУ методом Гаусса

private int SolveMatrix()

{

if (rowCount != columCount)

return 1; //нет решения

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

{

SortRows(i);

for (int j = i + 1; j < rowCount; j++)

{

if (matrix[i, i] != 0) //если главный элемент не 0, то производим вычисления

{

double MultElement = matrix[j, i] / matrix[i, i];

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

matrix[j, k] -= matrix[i, k] \* MultElement;

rightPart[j] -= rightPart[i] \* MultElement;

}

//для нулевого главного элемента просто пропускаем данный шаг

}

}

//ищем решение

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

{

answer[i] = rightPart[i];

for (int j = (rowCount - 1); j > i; j--)

answer[i] -= matrix[i, j] \* answer[j];

if ((matrix[i, i] >= -0.00001) && (matrix[i, i] <= 0.00001))

if ((rightPart[i] >= -0.00001) && (rightPart[i] <= 0.00001))

return 2; //множество решений

else

return 1; //нет решения

answer[i] /= matrix[i, i];

}

return 0;

}

}

}

**Вывод:**

Для проверки работоспособности кода были проведены решения таких наборов коэффициентов, что:

* определитель матрицы равнялся 0
* на выходе должны были получиться целые числа
* на выходе должны били получиться дробные числа
* система имела множество решений
* система не имела решений

Метод Гаусса является хорошим методом для решения СЛАУ, причем довольно прост в реализации. Однако для уменьшения погрешностей необходимо менять строки местами, чтобы в вершине находился максимальный элемент, за счет чего увеличивается время работы алгоритма. Также система должна быть совместна.

Вычисление определителя матрицы при размерности более чем 10 нежелательно, т.к. занимает очень продолжительное время.