Санкт-Петербургский государственный университет

**Курсовая работа**

по дисциплине «Численные методы»

на тему: «Метод Гаусса»

Студент: Марусина Анастасия Петровна

Группа: 209

Преподаватель: Перегудин Сергей Иванович

Метод Гаусса

Методы Гаусса (или методы исключения неизвестных) оптимальны для решения СЛАУ общего вида по количеству арифметических операций, необходимых для нахождения решения этой системы.

Простой метод Гаусса состоит в следующем: в СЛАУ ,

записанной в координатной форме, производится деление первого уравнения системы на коэффициент при первом неизвестном, после чего это преобразованное уравнение, будучи умноженным последовательно на все коэффициенты при первом неизвестном из всех нижележащих уравнениях системы, вычитается из соответствующих уравнений. Тем самым первое неизвестное оказывается “исключенным” из всех уравнений системы, кроме первого. Оставив на время в покое первое уравнение, таким же образом исключаем из системы уравнений с номерами 2-n второе неизвестное и так далее. Результатом проделанной работы явится уравнение, содержащее лишь последнее неизвестное системы , из которого оно и находится. Этот этап преобразования СЛАУ носит название прямого хода метода Гаусса. Вслед за найденным неизвестным находятся и все остальные. Этот этап называется обратным ходом метода Гаусса.

Отметим очевидный недостаток простого метода Гаусса – возможное обращение в нуль ведущих элементов, т.е. тех элементов, на которые приходится делить в прямом ходе метода Гаусса. Способ избавления от этого недостатка простого метода Гаусса – перестановка уравнений системы, которая, очевидно, не меняет решения СЛАУ. Ввиду того, что решается СЛАУ с неособой матрицей, при исключении первого неизвестного в первом столбце матрицы A найдётся нулевой элемент и строка, содержащая этот элемент, переставляется на место первого уравнения. С целью минимизировать погрешности округления обычно выбирают не просто ненулевой элемент в столбце, а элемент, максимальный по модулю и этот процесс повторяется на каждом этапе прямого метода Гаусса. Данная модификация называется методом Гаусса с выбором максимального элемента по столбцу.

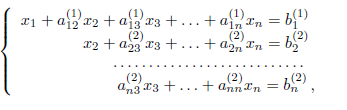
В формульном виде Метод Гаусса выглядит так: имеется СЛАУ, записанная в виде:

Пусть a11 ≠ 0, иначе меняем порядок уравнений с тем, чтобы было выполнено данное условие. Разделим первое уравнение на a11 , после чего оно примет вид

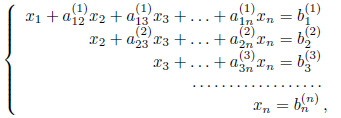
Где ,

Умножим уравнение (1.2) на a21 и вычтем полученное уравнение из второго уравнения системы (1.1). Аналогично преобразуем остальные уравнения. В итоге получим СЛАУ:

Теперь, оставив без изменений первое уравнение системы (1.3), можно применить описанную процедуру к системе из (n- 1)-го уравнений, исключив неизвестное x2 из третьего и последующих уравнений. Получим систему вида



Продолжая аналогичные вычисления, в итоге получим эквивалентную исходной треугольную систему



Откуда и находятся последовательно все компоненты решения.

Разновидностью метода Гаусса является метод Гаусса-Жордана, в котором совмещены прямой и обратный ход, т.е. начиная со второго шага метода Гаусса исключение неизвестных производится не только из нижележащих уравнений, но и из вышерасположенных. Все отмеченные разновидности метода требуют O(n3) арифметических операций и являются в этом смысле оптимальны-  
ми среди точных методов.

Реализация метода на языке C++ используя интегрированную среду разработки Visual Studio 2013:

#include "stdafx.h"

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

float Matrix[4][5] = {

{ 4.4, -2.5, 19.2, -10.8, 4.3 },

{ 5.5, -9.3, -14.2, 13.2, 6.8 },

{ 7.1, -11.5, 5.3, -6.7, -1.8 },

{ 14.2, 23.4, -8.8, 5.3, 7.2 } };

float sourceMatrix[4][5];

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

{

for (int j = 0; j < 5; j++)

{

cout << Matrix[i][j] << "\t";

sourceMatrix[i][j] = Matrix[i][j];

}

cout << endl;

}

//-------------

int column = 0;

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

{

float k = Matrix[line][column];

for (int i = column; i < 5; i++)

{

Matrix[line][i] = Matrix[line][i] / k;

}

for (int i = line + 1; i < 4; i++)

{

float multiplex = Matrix[i][column];

for (int j = column; j < 5; j++)

{

Matrix[i][j] = Matrix[i][j] - Matrix[line][j] \* multiplex;

}

}

column++;

}

cout << endl;

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

{

for (int j = 0; j < 5; j++)

{

cout << Matrix[i][j] << "\t";

}

cout << endl;

}

//------------

int column1 = 3;

for (int line1 = 3; line1 >= 0; line1--)

{

for (int i = line1 - 1; i >= 0; i--)

{

float x4 = Matrix[i][column1];

for (int j = 4; j >= 0; j--)

{

Matrix[i][j] = Matrix[i][j] - Matrix[line1][j] \* x4;

}

}

column1--;

}

//---------------

cout << endl;

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

{

for (int j = 0; j < 5; j++)

{

cout << Matrix[i][j] << "\t";

}

cout << endl;

}

float result[4];

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

{

cout << "x" << i << " = " << Matrix[i][4] << endl;

result[i] = Matrix[i][4];

}

//---------prove solution

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

{

float summator = 0;

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

{

summator += sourceMatrix[i][j] \* result[j];

}

if (summator - sourceMatrix[i][4] < 0.001)

{

cout << "success" << "\t" << "equation" << i + 1 << endl;

}

}

system("pause");

return 0;

}