Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по лабораторной работе
«Метод Гаусса на С++»

Выполнил:

студент группы 38200-1

Исаев В.И.

Проверил:

ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

Содержание:

Постановка задачи.

Цель моей работы заключалась в написании и изучении работы метода Гаусса для класса вектора векторов.

Метод решения

Рассмотрим алгоритм метода Гаусса:

1) Пусть в системе уравнений

$$\begin{cases} a_{11}^{(0)} \cdot x_1 + a_{12}^{(0)} \cdot x_2 + \ldots + a_{1m}^{(0)} \cdot x_m = f_1^{(0)} \\ a_{21}^{(0)} \cdot x_1 + a_{22}^{(0)} \cdot x_2 + \ldots + a_{2m}^{(0)} \cdot x_m = f_2^{(0)} \\ \ldots \\ a_{m1}^{(0)} \cdot x_1 + a_{m2}^{(0)} \cdot x_2 + \ldots + a_{mm}^{(0)} \cdot x_m = f_m^{(0)} \end{cases}$$

первый элемент a_{11} не равен 0. Назовем его ведущим элементом первой строки. Поделим все элементы этой строки на a_{11} и исключим x_1 из всех последующих строк, начиная со второй, путем вычитания первой (преобразованной), умноженной на коэффициент при x_1 в соответствующей строке. Получим

$$\begin{cases} x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + \ldots + a_{1m}^{(1)} \cdot x_m = f_1^{(1)} \\ \quad a_{22}^{(1)} \cdot x_2 + a_{23}^{(1)} \cdot x_3 + \ldots + a_{2m}^{(1)} \cdot x_m = f_2^{(1)} \\ \ldots \\ \quad a_{m2}^{(1)} \cdot x_2 + a_{m3}^{(1)} \cdot x_3 + \ldots + a_{mm}^{(1)} \cdot x_m = f_m^{(1)} \end{cases}$$

2) Если a_{22}, то, продолжая аналогичное исключение, приходим к системе уравнений с верхней треугольной матрицей

$$\begin{cases} x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + \ldots + a_{1m}^{(1)} \cdot x_m = f_1^{(1)} \\ \qquad x_2 + a_{23}^{(2)} \cdot x_3 + \ldots + a_{2m}^{(2)} \cdot x_m = f_2^{(2)} \\ \qquad\qquad x_3 + \ldots + a_{3m}^{(3)} \cdot x_m = f_3^{(3)} \\ \qquad\qquad \ldots \quad \ldots \qquad \ldots \\ \qquad\qquad\qquad x_m = f_m^{(m)} \end{cases}$$

Из нее в обратном порядке находим все значения x_i:

$$\begin{cases} x_m = f_m^{(m)} \\ x_{m-1} = f_{m-1}^{(m-1)} - a_{m-1m}^{(m-1)} \cdot x_m \\ \dots \quad \dots \quad \dots \\ x_1 = f_1^{(1)} - a_{12}^{(1)} \cdot x_2 - a_{13}^{(1)} \cdot x_3 - \dots - a_{1m}^{(1)} \cdot x_m \end{cases}$$

3) Процесс приведения к системе с треугольной матрицей называется прямым ходом, а нахождения неизвестных - обратным. Если один из ведущих элементов равен нулю, изложенный алгоритм метода Гаусса неприменим. Тем не менее, для нормальной матрицы с ненулевым определителем всегда возможна такая перестановка уравнений, что на главной диагонали не будет нулей. В приведенном коде для простоты перестановок не делается, зато делается проверка решения, а прямой и обратный ход для наглядности вынесены в отдельные подпрограммы.

Руководство пользователя.

1. Первое что встречает пользователя - запрос на ввод размерности будущей матрицы.

```
C:\Users\sexon\source\repos\GAUSS LABS\Debug\GAUSS LABS.exe
Dim matrix:
```

2. После ввода размерности, программа запрашивает разрешение на продолжение, после чего автоматически генерируется матрица заданной размерности.

```
Консоль отладки Microsoft Visual Studio
Dim matrix:
5
Show example?
Write yes/ok to continue.
Ok

matrix:
41    18467   6334    26500   19169   15724
11478   29358   26962   24464   5705    28145
23281   16827   9961    491     2995    11942
4827    5436    32391   14604   3902    153
292     12382   17421   18716   19718   19895
```

3. Затем сразу же выводится преобразованная матрица и ответ в столбцовой форме.

```
Matrix Gauss method:
1   0   0   0   0   -1.40622
0   1   0   0   0   2.29029
0   0   1   0   0   0.480829
0   0   0   1   0   -1.63597
0   0   0   0   1   0.71962

Solution Column:
 = -1.40622
 = 2.29029
 = 0.480829
 = -1.63597
 = 0.71962
```

Так-как целью работы являлось только изучение работы алгоритма - в программе реализована только случайная генерация коэффициентов.

Описание программной реализации.

1.

```
T1 MaxCh(size_t Ncol, size_t BottomLine) {
    T1 Max = this->vector[BottomLine].GetElement(BottomLine);
    size_t Max1 = BottomLine;
    for (size_t i = BottomLine; i < this->size - 1; i++)
        if (abs(Max) < abs(this->vector[i + 1].GetElement(Ncol))) {
            Max = this->vector[i + 1].GetElement(Ncol);
            Max1 = i + 1;
        }
    return Max1;
}
```

Получение максимального элемента столбца.

2.

```
void SwapEl(Matrix<T2>& A, size_t Nrow) {
    A.Swap(Nrow, A.MaxCh(Nrow, Nrow));
}
```

Постановка максимального элемента на главную диагональ

3.

```
template <typename T2>
void High(Matrix<T2>& A, size_t Nelement, size_t size) {
    A.GetVector(Nelement) /= A.printEL(Nelement, Nelement);
    Vector<T2> tmp = A.GetVector(Nelement);
    for (size_t i = Nelement; i > 0; i--) {
        if (Nelement - i < 0)
            continue;
        else {
            tmp *= A.printEL(Nelement, Nelement - i);
            A.GetVector(Nelement - i) -= tmp;
            tmp = A.GetVector(Nelement);
        }
    }
}

template <typename T2>
void Low(Matrix<T2>& A, size_t Nelement, size_t size) {
    A.GetVector(Nelement) /= A.printEL(Nelement, Nelement);
    Vector<T2> tmp = A.GetVector(Nelement);
    for (size_t i = 1; i < size - Nelement; i++) {
        if (Nelement + i > size - 1)
            continue;
        else {
            tmp *= A.printEL(Nelement, Nelement + i);
            A.GetVector(Nelement + i) -= tmp;
            tmp = A.GetVector(Nelement);
        }
    }
}
```

Исключение элементов выше и ниже главной диагонали соответственно.

Подтверждение корректности

Для подтверждения корректности сравним результат работы алгоритма с аналогичными, представленными в интернете:

1) Результат работы представленного алгоритма:

```
matrix:
1   7   4   0
9   4   8   8
2   4   5   5


Matrix Gauss method:
1   0   0   -0.46602
0   1   0   -1.12621
0   0   1   2.08738

Solution Column:
 = -0.46602
 = -1.12621
 = 2.08738
```

2) Результат работы алгоритма с сайта https://matworld.ru/calculator/gauss-method-online.php :

Найти общее решение неоднородной системы линейных уравнений:

$$\begin{cases} 1x_1 + 7x_2 + 4x_3 = 0 \\ 9x_1 + 4x_2 + 8x_3 = 8 \\ 2x_1 + 4x_2 + 5x_3 = 5 \end{cases}$$

Решение в векторном виде:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -0.46601 \\ -1.12621 \\ 2.08737 \end{bmatrix}$$

Заметим, что решение выполнено полностью корректно.

Проведем еще одну проверку, для больших чисел:

1) Пример работы алгоритма:

```
Dim matrix:
3
Show example?
Write yes/ok to continue.
Ok

matrix:
41   18467   6334   26500
19169   15724   11478   29358
26962   24464   5705   28145


Matrix Gauss method:
1   0   0   -0.112057
0   1   0   0.931406
0   0   1   1.46895

Solution Column:
 = -0.112057
 = 0.931406
 = 1.46895
Another example?
```

2) Результат работы алгоритма с сайта https://matworld.ru/calculator/gauss-method-online.php :

Решение в векторном виде:

$$x=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} -0.112056 \\ 0.931406 \\ 1.468947 \end{bmatrix}$$

Заметим, что в данном случае погрешность составила 8,924109373884486e-6 , что является крайне малым значением.

Заключение

В ходе работы я освоил работу с методом Гаусса, реализованным на языке C++. Стоит отметить, что для достижения большей точности можно установить большее количество знаков после запятой, или реализовать класс обыкновенных дробей.