**Министерство образования Республики Беларусь**

**Белорусский государственный университет**

Факультет прикладной математики и информатики

Лабораторная работа №1

По курсу «Вычислительные методы алгебры»

**Метод Гаусса**

**(схема единственного деления)**

Вариант №5

Работу выполнил:

студент 3 курса 7 группы

**Шатерник Артём**

Преподаватель:

**Будник А. М**.

**Минск 2023**

1. **Постановка задачи.**

Найти решение системы линейных алгебраических уравнений c расширенной матрицей вида

0.5757 -0.0758 0.0152 0.0303 0.1061 3.5148

0.0788 0.9014 0.0000 -0.0606 0.0606 3.8542

0.0455 0.0000 0.7242 -0.2121 0.1212 -4.9056

-0.0909 0.1909 0.0000 0.7121 -0.0303 2.3240

0.3788 0.0000 0.1364 0.0152 0.8484 0.1818

применяя метод Гаусса (схема единственного деления).

Найти обратную матрицу , число обусловленности, вектора невязок для решения первоначальной матрицы и для вычисления обратной матрицы.

1. **Алгоритм решения.**

Пусть дана система:

**Прямой ход.**

Приводим её к верхнетреугольному виду с помощью следующего алгоритма из итераций:

1. Выбираем элемент в качестве ведущего элемента и

делим i-ую строку на .

1. Исключаем из всех остальных уравнений, вычитая из них i-ое уравнение, умноженное на , где j – номер соответствующего уравнения.

Получим систему с верхнетреугольным видом

**Обратный ход.**

Получаем по формулам:

*.*

**Определитель матрицы вычисляется по формуле:**

т.е. он равен произведению ведущих элементов при каждой итерации.

**Нахождение обратной матрицы.**

Столбцы обратной матрицы вычисляются из системы , где

и – символ Кронекера.

**Невязки.**

Невязка для вектора решений:  **.**

Невязка для обратной матрицы: **.**

**Число обусловленности.**

Использовалась первая матричная норма:

1. **Листинг программы.**

Функция для метода Гаусса.

template <typename T>

std::vector < std::vector <T>> gaussian\_elimination(int size,

std::vector <std::vector <T>> a\_matrix,

std::vector <std::vector <T>> b\_vector) {

// Метод Гаусса

std::vector <std::vector <T>> x\_result(size, std::vector <T>(1));

long double det\_a = 1;

// Прямой ход

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

long double leading\_elem = a\_matrix[i][i];

// Проверить что ведущий элемент не близок к нулю

for (int k = i + 1; k < size; k++) {

if (abs(leading\_elem) < 0.00001) {

a\_matrix[i][i] = 0;

std::swap(a\_matrix[i], a\_matrix[k]);

std::swap(b\_vector[i], b\_vector[k]);

leading\_elem = a\_matrix[i][i];

}

else {

break;

}

}

// Вычисляем определитель

det\_a \*= leading\_elem;

b\_vector[i] = b\_vector[i] \* (1 / leading\_elem);

a\_matrix[i] = a\_matrix[i] \* (1 / leading\_elem);

for (int j = i + 1; j < size; j++) {

b\_vector[j] = b\_vector[j] - a\_matrix[j][i] \* b\_vector[i];

a\_matrix[j] = a\_matrix[j] - a\_matrix[j][i] \* a\_matrix[i];

a\_matrix[j][i] = 0;

}

}

// Обратный ход

for (int i = size - 1; i >= 0; i--) {

x\_result[i][0] = b\_vector[i][0];

for (int j = size - 1; j > i; j--) {

x\_result[i][0] -= x\_result[j][0] \* a\_matrix[i][j];

}

}

return x\_result;

}

// Первая матричная норма

template <typename T>

T first\_matrix\_norm(std::vector <std::vector <T>> matrix) {

int size = matrix.size();

T sum = 0;

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

for (int j = 0; j < size; j++) {

matrix[i][j] = abs(matrix[i][j]);

}

}

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

auto temp = matrix[i];

sum += \*std::max\_element(begin(temp), end(temp));

}

return sum;

}

// Произведение матриц

template <typename T>

std::vector <std::vector <T>> matrix\_product(std::vector <std::vector <T>> a\_matrix,

std::vector <std::vector <T>> b\_matrix) {

int res\_rows = a\_matrix.size();

int res\_columns = b\_matrix[0].size();

std::vector <std::vector <T>> result(res\_rows, std::vector <T>(res\_columns));

if (a\_matrix[0].size() == b\_matrix.size()) {

for (int i = 0; i < res\_rows; i++) {

for (int j = 0; j < res\_columns; j++) {

for (int k = 0; k < a\_matrix[0].size(); k++) {

result[i][j] += a\_matrix[i][k] \* b\_matrix[k][j];

}

}

}

}

else {

throw "incorret matrix size";

}

return result;

}

int main() {

// Ввод данных

int size = 5;

std::vector <std::vector <long double>> x\_result(size, std::vector <long double>(1));

std::vector <std::vector <long double>> a\_matrix(size, std::vector <long double> (size));

std::vector <std::vector <long double>> b\_vector(size, std::vector <long double>(1));

std::ifstream input("input.txt");

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

for (int j = 0; j < size; j++) {

input >> a\_matrix[i][j];

}

}

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

input >> b\_vector[i][0];

}

x\_result = gaussian\_elimination(size, a\_matrix, b\_vector);

// Обратная матрица

std::vector <std::vector <long double>> inv\_matrix(size, std::vector <long double>(size));

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

std::vector <std::vector <long double>> e\_vec(size, std::vector <long double>(1));

e\_vec[i][0] = 1;

std::vector <std::vector <long double>> inv\_column = gaussian\_elimination(size, a\_matrix, e\_vec);

for (int j = 0; j < size; j++) {

inv\_matrix[j][i] = inv\_column[j][0];

}

}

// Вычисление невязок

std::vector <std::vector <long double>> r = matrix\_product(a\_matrix, x\_result) - b\_vector;

std::vector <std::vector <long double>> r\_inv = matrix\_product(inv\_matrix, a\_matrix);

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

r\_inv[i][i] -= 1;

}

// Число обусловленности

long double u = first\_matrix\_norm(a\_matrix) \* first\_matrix\_norm(inv\_matrix);

return 0;

}

1. **Результат и его анализ.**

A в верхнетреугольном виде:

1 -0.1317 0.0264 0.0526 0.1843 |6.1053

0 1 -0.0023 -0.071 0.0505 |3.6995

0 0 1 -0.2961 0.1556 |-7.1998

0 0 0 1 -0.0315 |3.0629

0 0 0 0 1 |-2.0007

x = ( 7.0012 3.9999 -6.0003 2.9999 -2.0007)

det A = 0.209991

A^-1 =

1.8679 0.1692 0.0077 -0.0575 -0.2488

-0.0907 1.0803 0.014 0.1013 -0.0642

0.0917 -0.0802 1.4153 0.4149 -0.1931

0.2265 -0.2705 -0.0126 1.367 0.0416

-0.8528 -0.0578 -0.2307 -0.0655 1.3201

Невязка r = Ax - b:

( -4.44089e-16 -8.88178e-16 -8.88178e-16 0 -2.498e-16 )

Норма невязки r = 2.47025e-15

Невязка r\_inv = (A^-1)A - E:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | -3.64292e-17 | 0 | 2.12504e-17 | 2.77556e-17 |
| -6.93889e-18 | 0 | -1.73472e-18 | -5.42101e-19 | 0 |
| 0 | 0 | -2.22045e-16 | -3.72966e-17 | 0 |
| -6.93889e-18 | 0 | 8.67362e-19 | -1.11022e-16 | 0 |
| 0 | 1.56125e-17 | 2.77556e-17 | 3.46945e-18 | 2.22045e-16 |

Норма невязки r\_inv = 5.9848e-16

Число обусловленности:

u = 26.52

Нормы невязок для решения системы и обратной матрицы малы, число обусловленности также небольшое. Можно сказать что полученное решение близко к реальному решению.