**Министерство образования Республики Беларусь**

**Белорусский государственный университет**

Факультет прикладной математики и информатики

Лабораторная работа №2

По курсу «Вычислительные методы алгебры»

**Метод отражений**

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 отражений. Вычислить невязки и сравнить с методом Гаусса по точности и экономичности.

1. **Алгоритм решения.**

Для метода необходимо построение матрицы отражений (Хаусхолдера) вида:

, где вектор единичной длины в сферической норме. Если размерность матрицы , то метод состоит из итерации.

k-ая итерация метода выглядит следующим образом:

* Берём вектор и вектор в котором на k-ом месте стоит единица, а остальные координаты равны нулю
* Строим вектор по формулам:

**, , .**

* Формируем матрицу отражений по формуле выше.
* Умножаем матрицу из прошлой итерации слева на матрицу отражений: **.**

После аналогично методу Гаусса производиться обратный ход:

***.***

1. **Листинг программы.**

// Транспонирование

template <typename T>

std::vector <std::vector<T>> transpose(std::vector <T> vec) {

int size = vec.size();

std::vector <std::vector<T>> result(size, std::vector <T>(1));

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

result[i][0] = vec[i];

}

return result;

}

// Скалярное произведение

template <typename T>

T scalar\_product(std::vector<T>& a, std::vector <T>& b) {

T result = 0;

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

result += a[i] \* b[i];

}

return result;

}

// Метод отражений

template <typename T>

std::vector < std::vector <T>> householder\_method(int size,

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

std::vector <std::vector <T>> b\_vector, bool show\_info = 0) {

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

// Прямой ход

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

// Вычисление w

std::vector <T> s(size);

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

s[k] = a\_matrix[k][i];

}

T alpha = sqrt(scalar\_product(s, s));

auto s\_temp = s;

s\_temp[i] -= alpha;

std::vector <std::vector <T>> w(1, std::vector<T>(size));

w[0] = s\_temp \* (1 / sqrt(2 \* scalar\_product(s, s\_temp)));

// Вычисление V

std::vector <std::vector <T>> V = matrix\_product(transpose(w[0]), w);

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

V[i] = V[i] \* (-2);

V[i][i] += 1;

}

a\_matrix = matrix\_product(V, a\_matrix);

b\_vector = matrix\_product(V, b\_vector);

}

// 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];

}

x\_result[i][0] /= a\_matrix[i][i];

}

// Вывод данных в консоль

if (show\_info) {

std::cout << "A в верхнетреугольном виде:" << std::endl;

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

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

std::cout << std::setw(10) << round(a\_matrix[i][j] \* 10000) / 10000 << std::setw(10);

}

std::cout << '|' << round(b\_vector[i][0] \* 10000) / 10000 << std::endl;

}

std::cout << std::endl << "x = (";

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

std::cout << std::setw(8) << round(x\_result[i][0] \* 10000) / 10000 << std::setw(8);

}

std::cout << std::setw(1) << ")" << std::endl;

}

return x\_result;

}

int main() {

setlocale(LC\_ALL, "Russian");

// 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];

}

// Вызов метода

x\_result = householder\_method(size, a\_matrix, b\_vector, 1);

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

std::cout << std::endl << "Невязка r = Ax - b:" << std::endl << "( " << std::setw(5);

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

std::cout << std::setw(14) << r[i][0] << std::setw(14);

}

std::cout << std::setw(5) << ")" << std::endl;

std::cout << std::endl << "Норма невязки r = " << first\_matrix\_norm(r) << std::endl;

return 0;

}

1. **Результат и его анализ.**

A в верхнетреугольном виде:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0.701 | 0.0143 | 0.1332 | -0.0798 | 0.5642 | |2.7981 |
| 0 | 0.9244 | -0.0033 | 0.0867 | 0.0354 | |3.9067 |
| 0 | 0 | 0.7249 | -0.1933 | 0.1794 | |-5.2889 |
| 0 | 0 | 0 | 0.7111 | 0.0588 | |2.0155 |
| 0 | 0 | 0 | 0 | 0.6286 | |-1.2576 |

x = ( 7.0012 3.9999 -6.0003 2.9999 -2.0007)

Невязка r = Ax - b:

(1.33227e-15 0 -8.88178e-16 -8.88178e-16 8.60423e-16)

Норма невязки r = 3.96905e-15

Сравнение метода Гаусса и метода отражений.

Экономичность:

Оба метода имеют сложность , но метод отражений требует в 2 раза больше умножений, которые являются более требовательными операциями. Можно сделать вывод что метод Гаусса является более эффективным. При этом метод отражений имеет плюс – он не меняет число обусловленности, а значит может решать большее число задач.

Точность:

Кубическая норма для метода Гаусса составила .

Для метода отражений она составила .

Их порядок совпадает, так как оба метода являются точными и приближённые значения мы получаем только из-за округления.

В данном случае более точным оказался метод Гаусса.