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 $Ax = b$ с расширенной матрицей вида

A					b
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.

2. Алгоритм решения.

Для метода необходимо построение матрицы отражений (Хаусхолдера) вида: $\boldsymbol{V = E - 2\omega\omega^T}$, где ω вектор единичной длины в сферической норме. Если n размерность матрицы А, то метод состоит из $n - 1$ итерации.

k-ая итерация метода выглядит следующим образом:

- Берём вектор $\boldsymbol{s^k = (0, \dots, a_{kk}^{k-1}, \dots, a_{nk}^{k-1})^T}$ и вектор $\boldsymbol{e^k = (0, \dots, 0, 1, 0, \dots, 0)}$ в котором на k-ом месте стоит единица, а остальные координаты равны нулю
- Строим вектор ω по формулам:

$$\boldsymbol{\alpha = \sqrt{(s,s)}, \quad k = \frac{1}{\sqrt{2(s, s - \alpha e)}}, \quad \omega = k(s - \alpha e).}$$

- Формируем матрицу отражений по формуле выше.
- Умножаем матрицу A^{k-1} из прошлой итерации слева на матрицу отражений: $A^{\boldsymbol{k}} = \boldsymbol{VA^{k-1}}$.

После аналогично методу Гаусса производиться обратный ход:

$$\boldsymbol{x_n = \frac{q_n}{a_{nn}}}$$

$$\boldsymbol{x_i = q_i - \sum_{j=i+1}^{n} a_{ij}\, x_j\,, \ \ i = \overline{n-1, 1}\,.}$$

3. 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;
}
```

4. 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 $O(n^3)$, 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 $r_1 = 2.47025\text{e} - 15$.
Для метода отражений она составила $r_2 = 3.96905\text{e} - 15$.
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.