МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №3

По курсу «Вычислительные методы алгебры»

Метод Зейделя

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

2. Алгоритм решения.

Предварительные преобразования.

Для использования метода Зейделя систему уравнений нужно привести к каноническому виду $x = Bx + g$.

Чтобы построить матрицу B для начала приведём матрицу A к симметричному виду. Воспользуемся трансформацией Гаусса:

- Вычислим A^T.
- Умножим систему слева на A^T: $\mathrm{A}^T Ax = \mathrm{A}^T b$.
- Полученная матрица $\bar{A} = \mathrm{A}^T A$ будет симметричной, а система будет иметь вид $\bar{A}x = \bar{b}$

Матрицу B будем строим по формуле $B = E - C\bar{A}$, при этом $C = \frac{1}{||A||}$.

При таком построении собственные значения матрицы B будут меньше единицы и метод Зейделя будет сходиться по необходимому и достаточному условию.

Метод Зейделя.

$$x_i^{(k+1)} = \sum_{j=1}^{i-1} b_{ij}\, x_j^{(k+1)} + \sum_{j=1}^{i-1} b_{ij}\, x_j^{(k)} + g_i\,, \qquad i = 1, \dots, n\,;\ \ k = 0, 1, \dots$$

Критерий остановки процесса: $\left\| x^{k+1} - x^k \right\| \le \varepsilon$, где $\varepsilon = 10^{-5}$.

3. Листинг программы.

```
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];
	}
	// Приводим матрицу к симметричному виду
	// Симметричный вид
	auto b_vector_sim = matrix_product(transpose(a_matrix), b_vector);
	auto a_matrix_sim = matrix_product(transpose(a_matrix), a_matrix);
	long double c = 1 / first_matrix_norm(a_matrix_sim);
	// B = E - c * A_sim
	std::vector <std::vector <long double>> b_matrix(size, std::vector <long double>(size));
	for (int i = 0; i < size; i++) {
		b_matrix[i] = -c * a_matrix_sim[i];
		b_matrix[i][i] = 1 + b_matrix[i][i];
	}
	// g = c * b_sim
	std::vector <std::vector <long double>> g_vector(size, std::vector <long double>(1));
	g_vector = c * b_vector_sim;
	// Точность
	long double e = 1e-5;
	// Метод Зейделя
	x_result = b_vector;
	int i = 0;
	auto x_result_new = x_result;
	while(true) {
		i++;
		for (int i = 0; i < size; i++) {
			long double sum = 0;
			for (int j = 0; j < size; j++) {
				sum += b_matrix[i][j] * x_result_new[j][0];
			}
			x_result_new[i][0] = sum + g_vector[i][0];
		}
		if (first_matrix_norm(x_result - x_result_new) <= e) {
			break;
		} else {
			x_result = x_result_new;
		}
```

```
        }
        // Вывод данных
        std::cout << "x = (";
        for (int i = 0; i < size; i++) {
                std::cout << std::setw(8) << round(x_result_new[i][0] * 10000) / 10000 << std::setw(8);
        }
        std::cout << std::setw(1) << ")" << std::endl;

        std::cout << "Колличество итераций: " << i << std::endl;
        std::vector <std::vector <long double>> r = matrix_product(a_matrix, x_result_new) - 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.

x = (7.0011 3.9999 -6.0003 2.9998 -2.0007)

Количество итераций: 65

Невязка r = Ax - b:

(-2.45592e-05 -1.66302e-06 -6.71944e-06 -2.94597e-06 9.22789e-06)

Норма невязки r = 2.455922e-05 (первая норма)

Экономичность:

Сходится со скоростью геометрической прогрессии. При данном методе построения матрицы B её норма получилась больше единицы, поэтому чтобы определить знаменатель этой прогрессии потребуются формулы использующие собственные значения, которые не приводились на лекциях.

Точность:

Метод является итерационным и даёт наперёд заданную точность. Ценой большей точности является большее количество итераций и соответственно большее количество операций. Однако, как и при методах Гаусса и отражений мы не можем получить точность выше 10^{-15} без изменения типа хранимых данных.