МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №4

По курсу «Вычислительные методы алгебры»

Метод минимальных невязок

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 минимальных невязок. Вычислить невязки и сравнить с другими методами по точности и экономичности.

2. **Алгоритм решения.**

Метод минимальных невязок применяется к симметрическим матрицам, поэтому 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 $\bar{A}x = \bar{b}$

Метод является итерационным и применяется до достижения наперёд заданной точности $\varepsilon = 10^{-5}$, т.е. в качестве критерия остановки процесса используем $\left\|x^{k+1} - x^k\right\| \le \varepsilon$.

Метод минимальных невязок.

Итерация выглядит следующим образом

$$x^{k+1} = x^k - \tau_{k+1} * r^k ,$$

где

$$\tau_{k+1} = \frac{(Ar^k, r^k)}{(Ar^k, Ar^k)}.$$

r^k – представляет собой невязку, которая вычисляется по формуле

$$r^{k+1} = r^k - \tau_{k+1} A r^k.$$

При этом $r^0 = Ax^0 - b$.

В качестве начального приближения x^0 возьмём значения вектора b.

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
	auto b_vector_sim = matrix_product(transpose(a_matrix), b_vector);
	auto a_matrix_sim = matrix_product(transpose(a_matrix), a_matrix);
	// Точность
	long double e = 1e-5;
	x_result = b_vector_sim;
	auto x_result_new = x_result;
	// Невязка
	std::vector <std::vector <long double>> residual;
	int i = 0;
	while (true) {
		i++;
		for (int i = 0; i < size; i++) {
			residual = matrix_product(a_matrix_sim, x_result) - b_vector_sim;
			std::vector <long double> ar =
transpose_to_vector(matrix_product(a_matrix_sim, residual));
			auto tau = scalar_product(ar, transpose_to_vector(residual)) / scalar_product(ar,
ar);
			for (int j = 0; j < size; j++) {
				residual[j] = residual[j] * (tau);
				x_result_new[j] = x_result[j] - residual[j];
			}
		}
		if (first_matrix_norm(x_result - x_result_new) <= e) {
			x_result = x_result_new;
			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 << "Количество 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 << std::scientific << "Норма невязки r = " << first_matrix_norm(r) <<
std::endl;


    return 0;
}
```

4. Результат и его анализ.

x = (7.0012 3.9999 -6.0003 2.9999 -2.0007)

Количество итераций: 36

Невязка r = Ax - b:

(-7.43123e-06 -6.09909e-07 -2.0983e-06 -7.66485e-07 2.51365e-06)

Норма невязки r = 7.431235e-06 (первая норма).

Экономичность:

Метод сходится при $A = A^T > 0$. Так перед применением метода мы привели систему к диагональному виду. Одна итерация метода требует $O(n^2)$ операций. Получилось меньшее число итераций, чем в методе Зейделя $(36 < 65)$. Это связанно с тем, что скорость сходимости метода Зейделя зависит от матрицы B, используемой в нём.

Точность:

Метод является итерационным и даёт наперёд заданную точность.
Однако, большая точность приводит к увеличению числа операций.