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

1. **Алгоритм решения.**

Метод минимальных невязок применяется к симметрическим матрицам, поэтому воспользуемся трансформацией Гаусса:

* Вычислим .
* Умножим систему слева на : .
* Полученная матрица будет симметричной, а система будет иметь вид

Метод является итерационным и применяется до достижения наперёд заданной точности , т.е. в качестве критерия остановки процесса используем

Метод минимальных невязок.

Итерация выглядит следующим образом

где

При этом .

В качестве начального приближения возьмём значения вектора

1. **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;

}

1. **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 Так перед применением метода мы привели систему к диагональному виду. Одна итерация метода требует операций. Получилось меньшее число итераций, чем в методе Зейделя . Это связанно с тем, что скорость сходимости метода Зейделя зависит от матрицы , используемой в нём.

Точность:

Метод является итерационным и даёт наперёд заданную точность.

Однако, большая точность приводит к увеличению числа операций.