Белорусский Государственный Университет

Факультет Прикладной Математики и Информатики

Отчет

Лабораторная работа №3

«Решение СЛАУ методом прогонки»

Профессор кафедры вычислительной математики ФПМИ

Лиходед Николай Александрович

Студент 2 группы 2 курса

Сачек Илья Валерьевич

2019 год

**Постановка задачи**:

Необходимый для выполнения работы теоретический материал и формулы имеются в файле «Метод прогонки».

**Задание.** Разработать программу численного решения методом прогонки СЛАУ вида *Ay=f* (вида (1) в файле «Метод прогонки»).

Матрицу системы задать следующим образом:

*.*

Правую часть *f* задать умножением матрицы *A* на вектор *y=*(1, 2, ... , *N*+1): *f=Ay*.

Для вычислений выбрать параметры:

* *m* – номер в списке студенческой группы;
* *N*+1 – (порядок матрицы) одно из чисел в пределах от 10 до 12;
* *k* – номер студенческой группы.

Программно реализовать (в качестве языка программирования выбрать C или C++) вычисления для рассматриваемого примера. Для вычислений использовать тип float.

**Входные данные:**

Начальная матрица:

17 16 0 0 0 0 0 0 0 0 0 0 0

-2 19 17 0 0 0 0 0 0 0 0 0 0

0 -2 20 18 0 0 0 0 0 0 0 0 0

0 0 -2 21 19 0 0 0 0 0 0 0 0

0 0 0 -2 22 20 0 0 0 0 0 0 0

0 0 0 0 -2 23 21 0 0 0 0 0 0

0 0 0 0 0 -2 24 22 0 0 0 0 0

0 0 0 0 0 0 -2 25 23 0 0 0 0

0 0 0 0 0 0 0 -2 26 24 0 0 0

0 0 0 0 0 0 0 0 -2 27 25 0 0

0 0 0 0 0 0 0 0 0 -2 28 26 0

0 0 0 0 0 0 0 0 0 0 -2 29 27

0 0 0 0 0 0 0 0 0 0 0 -2 30

Вектор f:

49 87 128 173 222 275 332 393 458 527 600 677 366

**Листинг программы:**

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

#include <fstream>

float R(float x) {

int k = 100;

return round(x \* k) / (float)k;

}

float CountInfelicity(std::vector<float> x\_de\_ure, std::vector<float> x\_de\_facto) {

float sum\_x2 = 0, sum\_x = 0;

for (int i = 0; i < x\_de\_facto.size(); i++) {

sum\_x2 += x\_de\_facto[i] \* x\_de\_facto[i];

sum\_x += (x\_de\_facto[i] - x\_de\_ure[i]) \* (x\_de\_facto[i] - x\_de\_ure[i]);

}

std::ofstream fout("errors.txt");

fout << sum\_x / sum\_x2 << std::endl;

fout.close();

return sum\_x / sum\_x2;

}

void LogMatrix(std::vector<std::vector<float>> matrix) {

std::ofstream fout("logs.txt", std::ios::app);

for (const auto& vec : matrix) {

for (auto& el : vec) {

fout << el << '\t';

}

fout << std::endl;

}

fout << std::endl << std::endl;

fout.close();

}

void LogVector(std::vector<float> vector) {

std::ofstream fout("logs.txt", std::ios::app);

for (const auto& el : vector) {

fout << el << '\t';

}

fout << std::endl << std::endl;

fout.close();

}

std::vector<std::vector<float>> MultMatrix(std::vector<std::vector<float>> a, std::vector<std::vector<float>> b\_) {

std::vector<std::vector<float>> c(a.size());

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

c[i].resize(b\_[i].size());

for (int j = 0; j < b\_[i].size(); j++) {

c[i][j] = 0;

for (int k = 0; k < a[i].size(); k++)

c[i][j] += a[i][k] \* b\_[k][j];

}

}

return c;

}

void PrintMatrix(const std::vector<std::vector<float>>& matrix) {

for (const auto& vec : matrix) {

for (const auto& x\_ : vec) {

std::cout << std::setw(7) << std::setprecision(2) << std::left << x\_ << " ";

}

std::cout << std::endl;

}

}

void PrintVector(const std::vector<float>& vector) {

for (const auto& x\_ : vector) {

std::cout << std::setw(7) << std::setprecision(2) << std::left << x\_ << " ";

}

std::cout << std::endl;

}

class Matrix {

public:

~Matrix() {

matrix\_.clear();

}

// создание матрицы А

Matrix() {

srand(time(NULL));

// случайное количество элементов матрицы

int N = rand() % 3 + 10, m = 17, k = 2;

matrix\_.resize(N + 1);

matrix\_[0].resize(N + 1);

matrix\_[0][0] = m, matrix\_[0][1] = m - 1;

y\_.push\_back(1);

// заполняем матрицу А

for (int i = 1; i < N; i++) {

matrix\_[i].resize(N + 1, 0.0f);

matrix\_[i][i - 1] = -k;

matrix\_[i][i] = m + k + i - 1;

matrix\_[i][i + 1] = m + i - 1;

y\_.push\_back(i + 1);

}

matrix\_[N].resize(N + 1, 0.0f);

matrix\_[N][N - 1] = -k;

matrix\_[N][N] = m + k + N - 1;

y\_.push\_back(N + 1);

f\_ = MultMat(y\_, matrix\_);

}

std::vector<float> GetY() { return y\_; }

// функция умножения матриц

std::vector<float> MultMat(std::vector<float> a, std::vector<std::vector<float>> matrix) {

std::vector<float> c(matrix.size());

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

c[i] = 0;

for (int k = 0; k < matrix.size(); k++) {

c[i] += matrix[i][k] \* a[k];

}

}

return c;

}

std::vector<float> Solve() {

// вывод на экран и в файл

LogMatrix(matrix\_); LogVector(f\_);

std::cout << "Vector y we need to get: " << std::endl;

PrintVector(y\_);

std::cout << "Input matrix: " << std::endl;

PrintMatrix(matrix\_);

std::cout << std::endl;

std::cout << "Vector f: " << std::endl;

PrintVector(f\_);

std::cout << std::endl;

std::cout << std::endl;

// зануляем нижнюю диагональ

matrix\_[0][1] /= matrix\_[0][0]; f\_[0] /= matrix\_[0][0];

matrix\_[0][0] = 1;

for (int i = 1; i < matrix\_.size(); i++) {

float dev = R(matrix\_[i][i - 1] / matrix\_[i - 1][i - 1]);

matrix\_[i][i] -= R(matrix\_[i - 1][i] \* dev);

f\_[i] -= R(f\_[i - 1] \* dev);

matrix\_[i][i - 1] -= R(matrix\_[i - 1][i - 1] \* dev);

}

// вывод на экран и в файл

LogMatrix(matrix\_); LogVector(f\_);

std::cout << "Matrix after first step: " << std::endl;

PrintMatrix(matrix\_);

std::cout << "Vector f after first step: " << std::endl;

PrintVector(f\_);

// вычисляем вектор решений

std::vector<float> answers(matrix\_.size());

answers[answers.size() - 1] = R(f\_[f\_.size() - 1] / matrix\_[matrix\_.size() - 1][matrix\_.size() - 1]);

for (int i = matrix\_.size() - 2; i >= 0; i--) {

answers[i] = R((f\_[i] - matrix\_[i][i + 1] \* answers[i + 1]) / matrix\_[i][i]);

}

LogVector(answers);

return answers;

}

private:

std::vector<std::vector<float>> matrix\_;

std::vector<float> f\_, y\_;

};

int main() {

remove("logs.txt");

Matrix m;

std::cout << std::endl;

std::vector<float> answers = m.Solve();

std::cout << std::endl << "Vector of answers:" << std::endl;

PrintVector(answers);

std::cout << "Error is: " << CountInfelicity(m.GetY(), answers) << std::endl;

std::cout << std::endl << std::endl;

return 0;

}

**Выходные данные:**

Матрица после прогонки:

1 0.941176 0 0 0 0 0 0 0 0 0 0 0

0 20.88 17 0 0 0 0 0 0 0 0 0 0

0 0.08999 21.7 18 0 0 0 0 0 0 0 0 0

0 0 -0.05 22.62 19 0 0 0 0 0 0 0 0

0 0 0 0.04 23.71 20 0 0 0 0 0 0 0

0 0 0 0 -0.1 24.6 21 0 0 0 0 0 0

0 0 0 0 0 -0.03 25.68 22 0 0 0 0 0

0 0 0 0 0 0 0.05 26.76 23 0 0 0 0

0 0 0 0 0 0 0 -0.13 27.61 24 0 0 0

0 0 0 0 0 0 0 0 -0.07 28.68 25 0 0

0 0 0 0 0 0 0 0 0 0.009999 29.75 26 0

0 0 0 0 0 0 0 0 0 0 0.07999 30.82 27

0 0 0 0 0 0 0 0 0 0 0 -0.15 31.62

Вектор f:

2.88235 92.76 137.28 185.36 238.68 294.09 355.53 421.44 487.5 561.13 639.28 721.75 409.31

Ответ:

0.99 2.01 2.99 4.02 4.97 6.04 6.93 8.07 8.93 10.04 10.93 12.08 12.94

Относительная погрешность:

**Вывод:**

Данный алгоритм решает СЛАУ. Кроме того, он позволяет выполнить это с достаточно малой погрешностью, т.е. наше решение отличается от точного совсем на маленькие значения. Позволяет выполнить вычисление за меньшее количество времени.