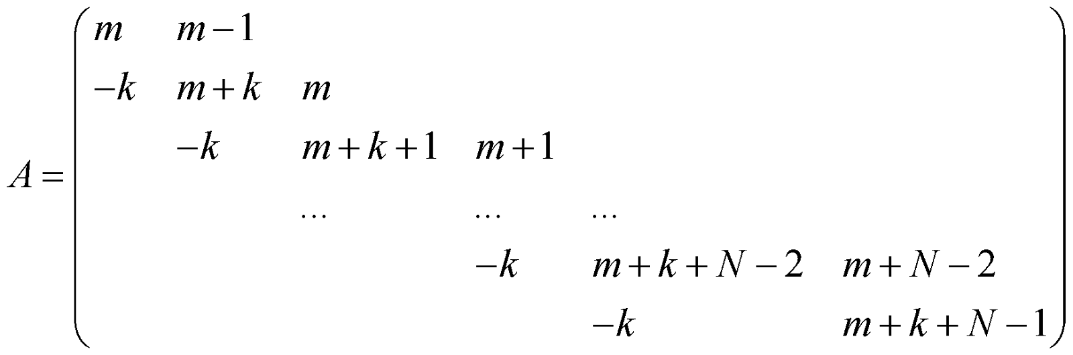
**Лабораторная работа №3**

**Задача.**

Разработать программу численного решения методом прогонки СЛАУ *Ay=f* вида. Матрицу системы задать следующим образом:

*.*

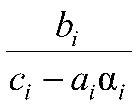
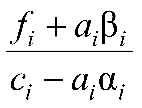
Правую часть *f* задать умножением матрицы *A* на вектор *y =* (1, 2, ..., *N*+1): *f=Ay*.

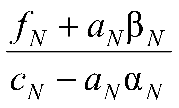
Для оценки погрешности вычислений использовалась кубическая норма.

Относительная погрешность , где – точное решение (*m*, *m*+1, ..., *n*+*m*–1).

Прямая прогонка – вычисление прогоночных коэффициентов по формулам:

α1*=**,* β1*=**,*

α*i+*1*=**,* β*i+*1*=**, i=* 1, 2, ..., *N*–1

β*N+*1*=**;*

Обратная прогонка – вычисление решения по формулам:

*yN*=β*N+*1, *yi=*α*i+*1*yi+*1*+*β*i+*1, *i=* *N*–1, ...,1, 0.

**Входные данные.**

N = 1800, m = 14, k = 2

**Листинг программы.**

#include <iostream>

#include <vector>

#include <stdlib.h>

#include <ctime>

#include <cmath>

#include <chrono>

#include <string>

int size = 4;

int m = 14;

int k = 2;

std::vector<float> generateVectorB() {

    std::vector<float>  ans(size+1, 0);

    for (size\_t i = 0; i < size+1; ++i) {

        ans[i] =i+1;

    }

    return ans;

}

std::vector<float> multiply(std::vector<std::vector<float>> A, std::vector<float> x) {

    std::vector<float> ans(size+1, 0);

    {

        for (int i = 0; i < size+1; i++) {

            float sum = 0;

            for (int j = 0; j < size+1; j++) {

                sum += A[i][j] \* x[j];

            }

            ans[i] = sum;

        }

    }

    return ans;

}

std::vector<std::vector<float>> generateMatrix(int size) {

    std::vector<std::vector<float>> A(size+1, std::vector<float>(size+1, 0));

    A[0][0] = m;

    A[0][1] = (m-1);

    for (int i = 1; i < size; i++){

        A[i][i-1] = -k;

        A[i][i] = m + k + (i - 1);

        A[i][i+1] = (m + (i-1));

    }

    A[size][size-1] = -k;

    A[size][size] = m + k + size - 1;

    return A;

}

std::vector<float> ForwardRunThrough(std::vector<std::vector<float>> &A, std::vector<float>& f)

{

    std::vector<float> z(size+1, 0);

    std::vector<float> y(size+1, 0);

    z[0] = -(A[0][1]) / A[0][0];

    y[0] = f[0] / A[0][0];

    for (int i = 1; i < size; i++)

    {

        z[i] = -(A[i][i+1]/(A[i][i] + A[i][i-1]\*z[i-1]));

        y[i] = (f[i] - A[i][i-1]\*y[i-1])/(A[i][i] + A[i][i-1] \* z[i-1]);

    }

    y[size] = (f[size] - A[size][size-1] \* y[size-1])/(A[size][size] + z[size-1]\*A[size][size-1]);

    std::vector<float> ans(size+1, 0);

    ans[size] = y[size];

    for (int i = size-1; i >= 0; i--)

    {

        ans[i] = z[i] \* ans[i+1] + y[i];

    }

    return ans;

}

void print5(std::vector<float> vector) {

    for (int i = 0; i < 5; i++)

        std::cout << vector[i] << " ";

    std::cout << "\n";

}

float relativeError(std::vector<float> x\_, std::vector<float> x) {

    float max\_1 = -1;

    for (int i = 0; i < size+1; i++) {

        max\_1 = std::max(std::abs(x\_[i] - x[i]), max\_1);

    }

    float max\_2 = -1;

    for (int i = 0; i < size+1; i++) {

        max\_2 = std::max(std::abs(x[i]), max\_2);

    }

    return max\_1 / max\_2;

}

int main()

{

    std::vector<std::vector<float>> matrix = generateMatrix(size);

    std::vector<float> x = generateVectorB();

    std::vector<float> b = multiply(matrix, x);

    auto start = std::chrono::steady\_clock::now();

    std::vector<float> ans = ForwardRunThrough(matrix, b);

    auto end = std::chrono::steady\_clock::now();

    const int time1 = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    std::cout << "5 cordinats of solution ";

    print5(ans);

    std::cout << "otnositelnaya pogreshnost " << relativeError(ans, x) << "\n";

    std::cout << "Runtime " << time1 << " ms\n";

    return 0;

}

**Выходные данные.**

5 cordinats of solution 1 2 3 4 5

otnositelnaya pogreshnost 1.35558e-006

Runtime 1 ms

**Выводы.**

Применение метода прогонки для работы с трехдиагональной матрицами дает очень значительно ускоряет вычисления.