**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №2

**Метод прогонки**

**Выполнил:**

Крученков Евгений Андреевич

студент 2 курса 9 группы,

специальность

“прикладная математика”

**Преподаватель:**

Ассистент кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 2-3

Листинг программы ---------------------------------------------------------------- 3-6

Результаты --------------------------------------------------------------------------- 6

Выводы ------------------------------------------------------------------------------- 7

**Постановка задачи**

Написать и отладить программу, реализующую метод прогонки для численного решения систем линейных алгебраических уравнений Ay = f, где A- трёхдиагональная квадратная матрица порядка N +1.

Найти решение системы линейных алгебраических уравнений с трехдиагональной матрицей A порядка 10 . Для проведения вычислительного эксперимента применять матрицу с диагональным преобладанием.

**Краткие теоретические сведения**

Рассматриваем систему вида

Алгоритм правой прогонки:

1. Прямой ход метода прогонки (прямая прогонка):
2. Обратный ход метода прогонки (обратная прогонка):

=

= +

Теорема (достаточное условие применения и устойчивости метода прогонки). Пусть коэффициенты системы удовлетворяют следующим условиям:

И пусть одно из последних трёх неравенств выполняется строго. Тогда метод прогонки применим и устойчив.

**Листинг программы**

**Header.h**

#pragma once

#include <iostream>

#include <stdio.h>

#include <iomanip>

#include <fstream>

#include <string>

#include <time.h>

using namespace std;

class TD\_Sist

{

private:

int n;

double\* A;

double\* B;

double\* C;

double\* Y;

double\* F;

double\* Y\_;

double pogr;

public:

int size()

{

return n;

}

void ShowPogr()

{

cout << scientific << pogr << fixed << showpoint << setprecision(3);

}

TD\_Sist(int k)

{

this->n = k - 1;

A = new double[n + 1];

B = new double[n + 1];

C = new double[n + 1];

Y = new double[n + 1];

Y\_ = new double[n + 1];

F = new double[n + 1];

srand(time(0));

for (int i = 1; i <= n; i++)

{

A[i] = (((double)rand() / RAND\_MAX) \* 2 - 1) \* 100;

}

for (int i = 0; i <= n - 1; i++)

{

B[i] = (((double)rand() / RAND\_MAX) \* 2 - 1) \* 100;

}

for (int i = 0; i <= n; i++)

{

Y[i] = i + 1;

Y\_[i] = 0;

if (i == 0)

{

C[0] = abs(B[0]) + 5 + (((double)rand() / RAND\_MAX) \* 5);

}

else if (i == n)

{

C[n] = abs(A[n]) + 5 + (((double)rand() / RAND\_MAX) \* 5);

}

else

{

C[i] = abs(A[i]) + abs(B[i]) + 5 + (((double)rand() / RAND\_MAX) \* 5);

}

}

F[0] = C[0] \* Y[0] - B[0] \* Y[1];

F[n] = C[n] \* Y[n] - A[n] \* Y[n-1];

for (int i = 1; i < n; i++)

{

F[i] = C[i] \* Y[i] - A[i] \* Y[i - 1] - B[i] \* Y[i + 1];

}

}

~TD\_Sist()

{

delete[] A;

delete[] B;

delete[] C;

delete[] Y;

delete[] Y\_;

delete[] F;

}

void Progonka()

{

//Прямая прогонка

double\* a, \* b;

a = new double[n + 1];

b = new double[n + 2];

a[1] = B[0] / C[0];

b[1] = F[0] / C[0];

for (int i = 1; i <= n-1; i++)

{

double zn = C[i] - A[i] \* a[i];

//if (abs(zn) < 0,00000001) throw "Знаменатель мал";

a[i + 1] = B[i] / zn;

b[i + 1] = (F[i] + A[i] \* b[i]) / zn;

}

b[n + 1] = (F[n] + A[n] \* b[n]) / (C[n] - A[n] \* a[n]);

//Обратная прогонка

Y\_[n] = b[n + 1];

for (int i = n - 1; i >= 0; i--)

{

Y\_[i] = a[i + 1] \* Y\_[i + 1] + b[i + 1];

}

//pogr

double maxY = abs(Y[0]), max\_Y = abs(Y[0] - Y\_[0]);

for (int i = 0; i <= n; i++)

{

if (max\_Y < abs(Y[i] - Y\_[i]))

{

max\_Y = abs(Y[i] - Y\_[i]);

}

if (maxY < abs(Y[i]))

{

maxY = abs(Y[i]);

}

}

pogr = max\_Y / maxY;

}

friend ostream& operator <<(ostream& os, TD\_Sist& M)

{

os << "Вектор А:" << endl << fixed << showpoint << setprecision(5);

for (int i = 1; i <= M.size(); i++)

{

os << M.A[i] << " ";

}

os << endl;

os << "Вектор B:" << endl;

for (int i = 0; i <= M.size() - 1; i++)

{

os << M.B[i] << " ";

}

os << endl;

os << "Вектор C:" << endl;

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

{

os << M.C[i] << " ";

}

os << endl;

os << "Точное решение Y:" << endl << dec;

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

{

os << M.Y[i] << " ";

}

os << endl;

os << "Вектор F:" << endl << fixed << showpoint << setprecision(5);

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

{

os << M.F[i] << " ";

}

os << endl;

os << "Полученное решение Y\_:" << endl << fixed << showpoint << setprecision(20);

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

{

os << M.Y\_[i] << " "<<endl;

}

os << "Относительная погрешность: " << scientific << M.pogr << endl << dec;

return os;

}

};

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "Header.h"

#include <fstream>

#include<time.h>

using namespace std;

ifstream fin;

ofstream fout;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "ru");

TD\_Sist A1(10);

cout << A1<<endl;

A1.Progonka();

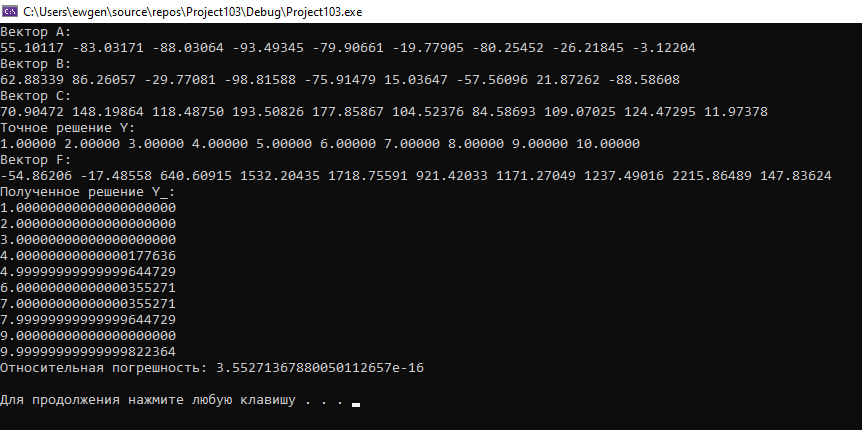
cout << A1<<endl;

system("pause");

return 0;

}

**Результаты**

****

**Выводы**

1. Метод прогонки является прямым методом, его точность  сопоставима с точностью метода Гаусса

2) По методу прогонки требуется произвести операций умножения и деления, т.е. трудоёмкость метода линейна, но метод прогонки можно применять только на трехдиагональных с диагональным преобладанием.

3) Также стоит отметить, что с помощью метода прогонки можно вычислить определитель матрицы:

det A =