БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №2

**Метод прогонки**

Выполнила:

Мастыкина Елизавета

2 курс 6 группа

Преподаватель:

Горбачева Ю.Н.

Минск, 2021

**Содержание**

Постановка задачи………………………………………………………....5

Алгоритм решения………………………………………………………....6

Листинг программы………………………………………………………..8

Вывод программы………………………………………………………….17

Выводы……………………………………………………………………..19

**Постановка задачи**

Написать и отладить программу, реализующую метод прогонки для численного решения систем линейных алгебраических уравнений Ay=f  с трехдиагональной матрицей A порядка N+1.Найти решение системы линейных алгебраических уравнений с трехдиагональной матрицей A порядка 10 .В результатах выполнения тестовой задачи необходимо привести следующую информацию: векторы a , c , b , точное решение y , вектор f , полученное численное решение y,  относительную погрешность.

**Алгоритм решения**

Пусть дана система

(1)

Алгоритм нахождения решения системы (1)

Прямой ход:

- прогоночные коэффициенты.

, i = .

Обратный ход:

,

, i = .

**Листинг программы**

#include<iostream>

#include<iomanip>

#include <ctime>

using namespace std;

void printMatrix(double\* matrix, int n)

{

//cout << "Matrix" << endl;

cout << endl;

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

{

cout << setw(30) << setprecision(25) << matrix[i]<< endl;

}

cout << endl;

}

void inputMatrix(double\* matrix, int n) {

cout << "Введите матрицу" << endl;

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

{

cin >> matrix[i];

}

}

void randomMatrix(double\* matrix, int n)

{

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

{

matrix[i] = -100 + rand() % (200 + 1);

}

}

// rand() % (b - a + 1) + a

void randomC(double\* c, double\* a, double\* b, int n)

{

srand(time(0));

c[0] = rand() % int(abs(b[0]) + 6 \* 2 - (abs(b[0]) + 6 + 1)) + abs(b[0]) + 6;

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

{

c[i] = rand() % int(abs(b[i]) + abs(a[i]) + 6 \* 2 - (abs(b[i])+ abs(a[i]) + 6 + 1)) + abs(b[i]) + abs(a[i]) + 6;

}

c[n-1] = rand() % int(abs(a[n-1]) + 6 \* 2 - (abs(a[n-1]) + 6 + 1)) + abs(a[n-1]) + 6;

}

void defineF(double\* f, double\* c, double\* a, double\* b,double\* exactSol, int n)

{

f[0] = c[0] \* exactSol[0] + b[0] \* exactSol[1];

f[n - 1] = a[n - 2] \* exactSol[n - 2] + c[n - 1] \* exactSol[n - 1];

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

{

f[i] = a[j] \* exactSol[i-1] + c[i] \* exactSol[i] + b[i] \* exactSol[i + 1];

}

}

double findAccuracy(double\* x, int n) {

double max = 0;

double accuracy;

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

{

if (abs(x[i]-(i+1)) > max)

{

max = abs(x[i] - (i + 1));

}

}

accuracy = max / n;

//cout << setprecision(40)<< max << endl;

return accuracy;

}

void main()

{

srand(time(0));

int n = 10;

double\* a = new double[n - 1];// ниже главной диагонали

double\* b = new double[n - 1];//выше главной диагонали

double\* c = new double[n];//на главной диагонали

double\* f = new double[n];//вектор правойчасти

double\* exactSol = new double[n];//вектор правойчасти

double\* x = new double[n];//столбец решений

double\* gama = new double[n];//на главной диагонали

double\* beta = new double[n];//на главной диагонали

double\* alpha = new double[n-1];//на главной диагонали

randomMatrix(a, n - 1);

randomMatrix(b, n - 1);

randomC(c, a, b, n);

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

{

exactSol[i] = i + 1;

}

defineF(f, c, a,b, exactSol, n);

cout << "a:\t";

printMatrix(a, n - 1);

cout << "b:\t";

printMatrix(b, n - 1);

cout << "c:\t";

printMatrix(c, n);

cout << "f:\t";

printMatrix(f, n);

cout << "exactSolution:\t";

printMatrix(exactSol, n);

// прямая прогонка

gama[0] = c[0];

alpha[0] = -b[0] / gama[0];

beta[0] = f[0] / gama[0];

for (size\_t i = 1; i < n-1; i++)

{

gama[i] = c[i] + a[i-1] \* alpha[i - 1];

alpha[i] = -b[i] / gama[i];

beta[i] = (f[i] - a[i-1] \* beta[i - 1]) / gama[i];

}

gama[n - 1] = c[n - 1] + a[n - 2] \* alpha[n - 2];

beta[n - 1] = (f[n - 1] - a[n - 2] \* beta[n - 2]) / gama[n - 1];

// обратная прогонка

x[n - 1] = beta[n - 1];

for (int i = n-2; i >=  0; i--)

{

x[i] = alpha[i] \* x[i + 1] + beta[i];

}

cout << "found Solution:\t";

printMatrix(x, n);

cout  << "Accuracy:" << setprecision(20) << findAccuracy(x, n) << endl;

}

**Результат вывода программы**

a:

                            14

                           -24

                            -4

                            29

                            71

                           -66

                           -85

                           -10

                           -36

b:

                            18

                            11

                           -55

                           -78

                            44

                            20

                            59

                             7

                           -31

c:

                            26

                            45

                            69

                           115

                           122

                            92

                           153

                            24

                            73

                             7

f:

                            62

                           137

                           -61

                            58

                           990

                          1047

                          1147

                          -340

                           267

                          -254

exactSolution:

                             1

                             2

                             3

                             4

                             5

                             6

                             7

                             8

                             9

                            10

found Solution:

   0.9999999999999997779553951

     2.00000000000000044408921

     2.99999999999999955591079

     3.99999999999999911182158

     4.99999999999999911182158

     6.00000000000000088817842

                             7

                             8

    9.000000000000001776356839

    10.00000000000000710542736

Accuracy:7.1054273576010022531e-16

**Выводы**

Метод прогонки является частным случаем метод Гаусса и используется для решения систем линейных уравнений вида Ax = B, где A — трёхдиагональная матрица.Если при решении таких систем применять метод Гаусса, то расчет можно организовать таким образом, чтобы не включать нулевые элементы матрицы. Этим самым экономится требуемая память и уменьшается объем вычислений. Указанное ускорение вычислений допускают системы линейных алгебраических уравнений с ленточными, блочными,квазитреугольными, почти треугольными и другими матрицами