Московский авиационный институт

(Национальный исследовательский университет)

Лабораторная работа №1

По курсу «Численные методы»

Тема « ЧИСЛЕННЫЕ МЕТОДЫ ЛИНЕЙНОЙ АЛГЕБРЫ»

Вариант №5

Выполнил: Цой В.Л.

Группа: 08-301

Проверил: Черкасов М.А.

Оценка:

Москва 2012

**Введение.**

Все программы реализована на языке С++ . Для реализации лабораторной работы использовалась кросс-платформенная среда разработки **Qt**.

Во всех частях этой лабораторной работы точкой входа в программу является функция *main*, которая описана в файле *main.cpp*. Все используемые функции в *main* объявлены в заголовочном файле *wid.h,* и в свою очередь описаны в файле исходных текстов *wid.cpp.*

Входные данные считываются из фала *input.txt(*либо из *inpu2t.txt, input3.txt* *)*, а все полученные результаты записываются в файл *otput.txt*.

**1.Метод LU разложения(метод Гаусса)**

* 1. main.cpp

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

/// переменные для

char name1[30];// Название файла для считывания

char name2[30];// Название файла для записи

char ok;// Пдтверждение для повторного выполнения программы

double x[N][N];// Матрица системы

double y[N];// Вектор правых частей системы

double l[N][N];// Матрица L

double u[N][N];// Матрица U

double p[N][N];// Матрица перестановок

double inv[N][N];// Обратная матрица

double temp[N][N];// Временный двумерный массив для проверки вычислений

double temp2[N];// Временный одномерный массив для проверки вычислений

double I[N];// Столбец еденичной матрицы

int index; // Количество перестановок

int okey;//Для зацикливания программы(повторного выполнения)

do

{

cout << "Please, enter the name of file for input matrix:\n";

cin >> name1;

cout << "Please, enter the name of file for output answer:\n";

cin >> name2;

index = 0.0;

InputMas(x, y, name1);

PLU(x, l, u, p, y, index);// Выполняем LU разложение

Determinant(u, index);// Вычисляем определитель

MultiMas(l, u, temp);// Проверяем,что произведение L на U дают

//исходную матрицу с точностью до перестановок строк

PLUinverse(u, y);//Находим корни из матрицы U

MultiMas(x, y, temp2);//Проверяем,что произведение исходной

//матрицы на вектор корней равен правой части

for(int i=0;i<N;i++)//Поиск обратной матрицы

{

NullMas(I);

I[i] = 1.0;

NullMas(l);

IdentityMas(l);

NullMas(u);

NullMas(p);

IdentityMas(p);

PLU(x, l, u, p, I, index);

PLUinverse(u, I);

for(int j=0;j<N;j++)

{

inv[j][i] = I[j];

}

}

NullMas(temp);

MultiMas(x, inv, temp);// Умножение исходной матрицы на

// обратную(должна получиться еденичная)

OutputAnswer(l, u, p, inv, y, name2);//Вывод результатов в файл

// Выводим результаты в консоль

cout << "\tMatrix L\n";

Test2(l);

cout << "\tMatrix U\n";

Test2(u);

cout << "\tMatrix P\n";

Test2(p);

cout << "\tMatrix Invers\n";

Test2(inv);

cout << "\tVector Y\n";

Test1(y);

cout << "\nDo you want the restart this programm ? (y/n)\n";

cin >> ok;

if (ok == 'y')

okey = 1;

else

okey = 0;

cout << "\n";

}

while(okey);

return a.exec();

}

* 1. wid.cpp

#include "wid.h"

void InputMas(double x[N][N], double y[N], char \*name1)

{

ifstream in;

in.open(name1);

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

{

for(int j=0;j<N;j++)

in >> x[i][j];

in >> y[i];

}

// if(N < maximum)

//PrintArr(arr\_input, x, y);

in.close();

}

void OutputAnswer(double l[N][N], double u[N][N], double p[N][N], double inv[N][N], double y[N], char \*name2)

{

ofstream out;

out.open(name2);

out << "Matrix L\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(l[i][j]) < minimum)

out << "0\t";

else

out << l[i][j] << "\t";

}

out << "\n";

}

out << "\nMatrix U\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(u[i][j]) < minimum)

out << "0\t";

else

out << u[i][j] << "\t";

}

out << "\n";

}

out << "\nMatrix P\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(p[i][j]) < minimum)

out << "0\t";

else

out << p[i][j] << "\t";

}

out << "\n";

}

out << "\nMatrix inverse\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(inv[i][j]) < minimum)

out << "0\t";

else

out << inv[i][j] << "\t";

}

out << "\n";

}

out << "\nAnswer\n";

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

{

if(abs(y[i]) < minimum)

out << "x" << i+1 << " = 0\n";

else

out << "x" << i+1 << " = " << y[i] << "\n";

}

out.close();

}

void Test2(double \_mas[N][N])//вывод на консоль 2х

{ //мерного массива

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

{

for(int j=0;j<N;j++)

{

if(abs(\_mas[i][j]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i][j]);

}

printf("\n");

}

printf("\n");

}

void Test1(double \_mas[N])//вывод на консоль одномерного массива

{

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

{

if(abs(\_mas[i]) < minimum )

printf("0.00\t");

else

printf("%.2f\t",\_mas[i]);

}

printf("\n\n");

}

void PLUinverse(double \_u[N][N], double \_y[N])

{

double a[ N ];

NullMas(a);

\_y[N-1] /= \_u[N-1][N-1];

for( int i = N-2; i > -1; i-- )

{

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

{

a[ i ] += \_u[ i ][ j ] \* \_y[ j ];

}

\_y[ i ] = ( \_y[ i ] - a[ i ]) / \_u [ i ][ i ];

}

}

void PLU(double \_x[N][N], double \_l[N][N], double \_u[N][N], double \_p[N][N], double \_y[N], int index)

{

Assignment(\_u, \_x);

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

{

//поиск главного элемента

double element = 0.0;

int number = -1.0;

for( int ind = i; ind < N; ind++ )

{

if( fabs(\_u[ ind ][ i ]) > element )

{

element = fabs(\_u[ ind ][ i ]);

number = ind;

index++;

}

}

if( element == 0 )

{

qDebug("The matrix is singular !");

}

//меняем местами i-ю строку и строку с главным элементом

SwapRows(\_p, number, i);// Для матрицы перестановок P

SwapRows(\_u, number, i);//Для матрицы U

SwapRows(\_y, number, i);// Для матрицы Y

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

{

\_l[ j ][ i ] = \_u[ j ][ i ] / \_u[ i ][ i ];

\_u[ j ][ i ] = 0.0;

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

{

\_u[ j ][ k ] -= \_l[ j ][ i ] \* \_u[ i ][ k ];

}

\_y [ j ] -= \_l[ j ][ i ] \* \_y[ i ];

}

}

}

void SwapRows(double \_mas[N], int \_i, int \_j)// замена строк местами

{

double temp;

temp = \_mas[\_i];

\_mas[\_i] = \_mas[\_j];

\_mas[\_j] = temp;

}

void SwapRows(double \_mas[N][N], int \_i, int \_j)// замена строк местами

{

double temp;

for(int k=0;k<N;k++)

{

temp = \_mas[\_i][k];

\_mas[\_i][k] = \_mas[\_j][k];

\_mas[\_j][k] = temp;

}

}

//присваивание массивов

void Assignment(double \_mas[N][N], double \_mas2[N][N])

{

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

{

for(int j=0;j<N;j++)

\_mas[i][j] = \_mas2[i][j];

}

}

//присваивание массивов

void Assignment(double \_mas[N], double \_mas2[N])

{

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

{

\_mas[i] = \_mas2[i];

}

}

void Determinant(double \_mas[N][N], int index)

{

double a = 1.0;

for(int j=0;j<index-N;j++)

{

a \*= (-1.0);

}

//qDebug("%d", index);

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

a \*= \_mas[i][i];

}

void NullMas(double \_mas[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas[i][j] = 0.0;

}

}

}

void NullMas(double \_mas[N])

{

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

{

\_mas[i] = 0.0;

}

}

void IdentityMas(double \_mas[N][N])//создание еденичной матрицы

{

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

{

\_mas[i][i] = 1.0;

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N])

{

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

{

for(int j=0;j<N;j++)

{

for(int k=0;k<N;k++)

\_mas3[i][j] += (\_mas[i][k])\*(\_mas2[k][j]);

}

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N])

{

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

{

for(int k=0;k<N;k++)

\_mas3[i] += (\_mas[i][k])\*(\_mas2[k]);

}

}

**1.3.**wid.h

#ifndef WID\_H

#define WID\_H

#include <QtCore/QCoreApplication>

//#include <QFile>

#include <stdio.h>

#include <cmath>

#include <fstream>

#include <iostream>

#include <stdlib.h>

using namespace std;

#define minimum 0.0000000001

#define N 4

//double x[N][N], y[N], l[N][N], u[N][N], p[N][N], inv[N][N], temp[N][N], temp2[N], I[N];

//int index;

//QString name1, name2;

void Test2(double \_mas[N][N]);/// Вывод на консоль двумерного массива

void Test1(double \_mas[N]);/// Вывод на консоль одномерного массива

void PLU(double \_x[N][N], double \_l[N][N], double \_u[N][N], double \_p[N][N], double \_y[N], int index);/// LU разложение с матрицей перестановок P

void PLUinverse(double \_u[N][N], double \_y[N]);/// Вычиляет корни после LU разложения

void SwapRows(double \_mas[N][N], int \_i, int \_j);/// Меняет местами строки в двумерном массиве

void SwapRows(double \_mas[N], int \_i, int \_j);/// Меняет местами элементы в одномерном массиве

void Assignment(double \_mas[N][N], double \_mas2[N][N]); /// Поэлементное присвание двумерных массивов

void Assignment(double \_mas[N], double \_mas2[N]);/// Поэлементное присванивание одномерных массивов

void Determinant(double \_mas[N][N], int index);/// Подсчёт определителя матрицы после LU разложения

void NullMas(double \_mas[N][N]);/// Обнуление двумерного массива

void NullMas(double \_mas[N]);/// Обнуление одномерного массива

void IdentityMas(double \_mas[N][N]);/// Создание еденичной матрицы

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N]);/// Перемножение матриц

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N]);/// Перемножение матрицы на вектор столбец

void InputMas(double x[N][N], double y[N], char \*name1);/// Считывание данных из файла

void OutputAnswer(double l[N][N], double u[N][N], double p[N][N], double inv[N][N], double y[N], char \*name2);/// Вывод данных(ответов) в файл

#endif // WID\_H

**1.4.** Входные данные(файл input3.txt)

3 -8 1 -7 96

6 4 8 5 -13

-1 1 -9 -3 -54

-6 6 9 -4 82

**1.5.**Выходные данные(файл otput.txt)

Matrix L

1 0 0 0

0.5 1 0 0

-0.166667 -0.166667 1 0

-1 -1 -0.583333 1

Matrix U

6 4 8 5

0 -10 -3 -9.5

0 0 14 -8.5

0 0 0 -8.70833

Matrix P (матрица перестановок)

0 1 0 0

1 0 0 0

0 0 0 1

0 0 1 0

Matrix inverse (обратная матрица)

0.062201 0.1311 0.101982 -0.0214627

-0.0239234 0.0880383 0.130007 0.0544087

0.0191388 0.00956938 -0.0697198 0.0307587

-0.0861244 -0.0430622 -0.114833 -0.0669856

Answer (ответ)

x1 = -3

x2 = -6

x3 = 8

x4 = -7

**2.Метод прогонки**

**2.1.**main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

#define N 4

int main(int argc, char \*argv[])

{

QCoreApplication asd(argc, argv);

double a[N],b[N],c[N],f[N],x[N];

char namefile[10]; // Имя файла считывания

char out\_ans[12]; // Имя файла записи

cout <<"Please, input name of file for Read:\n";

cin >> namefile;

Nullmas(a,b,c,f);

read\_of\_file(a,b,c,f, namefile);

cout <<"Please, input name of file for write:\n";

cin >> out\_ans;

if(diagPrevalence(a,b,c) == 0) // Проверка на диагональное преобладание

{

Prorace(a,b,c,f,x); // Выполнение метода прогонки

out\_ansver( x, out\_ans); // Вывод результата в фаил

}

else

printf("Not diagonal prevaletion\n");

printf("%d\n\n",diagPrevalence(a,b,c));

printarr(x); // Вывод результата на консоль

return asd.exec();

}

**2.2.**wid.h

#ifndef WID\_H

#define WID\_H

#include <fstream>

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include "stdlib.h"

#include "math.h"

using namespace std;

#define N 4

void read\_of\_file(double a[N],double b[N],double c[N],double f[N], char\* \_namefile); /// Считывание из файла

void printarr(double x[N]); /// Вывод на консоль

void Prorace(double \*a,double \*b,double \*c,double \*f, double \*x); /// Метод прогонки

void Nullmas(double a[N],double b[N],double c[N],double f[N]); /// Обнуление массивов

void out\_ansver(double x[N], char \*out\_ans); /// Вывод результата в фаил

int diagPrevalence(double a[N],double b[N],double c[N]); /// Проверка на диагональное преобладание

#endif // WID\_H

**2.3.**wid.cpp

#include "wid.h"

using namespace std;

void read\_of\_file(double a[N],double b[N],double c[N],double f[N], char\* \_namefile)//считать из файла

{

ifstream in;

in.open(\_namefile);

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

{

if(i==N-1)

{

in >> c[i];

in >> f[i];

}

else

{

in >> c[i];

in >> b[i];

in >> f[i];

in >> a[i+1];

}

}

in.close();

}

void printarr(double x[N])

{

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

{

printf("x%d = %f\n", i, x[i]);

}

}

void Prorace(double \*a,double \*b,double \*c,double \*f, double \*x)

{

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

{

double m =a[i]/c[i-1];

c[i]=c[i]-m\*b[i-1];

f[i]=f[i]-m\*f[i-1];

}

x[N-1]=f[N-1]/c[N-1];

for(int i=N-2;i>=0;i--)

{

x[i]=(f[i]-b[i]\*x[i+1])/c[i];

}

}

void Nullmas(double a[N],double b[N],double c[N],double f[N])

{

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

{

a[i]=0;

b[i]=0;

c[i]=0;

f[i]=0;

}

}

void out\_ansver(double x[N], char \*out\_ans)

{

ofstream out;

out.open(out\_ans);

out << "ansver\n";

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

{

out<<"x"<<i+1<<"="<<x[i]<<"\n";

}

out.close();

}

int diagPrevalence(double a[N],double b[N],double c[N])

{

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

{

if (abs(c[i]) < abs(b[i])+abs(a[i]))

return 1;

}

return 0;

}

**2.4.** Входные данные(файл input3.txt)

8 4 48

-5 22 8 125

-5 -11 1 -43

-9 -15 1 18

1 7 -23

**2.5.** Выходные данные(файл otput.txt)

ansver

x1=3.02184

x2=5.95632

x3=1.13377

x4=-0.746929

**3.Метод простых итераций**

**3.1.**main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication asd(argc, argv);

double y[N], a[N], b[N], temp1[N], x[N][N], x1[N][N];

int index; // Счётчик числа итераций

double accuracy = 0.001; // Заданная точность вычислений

QString name1 = "input.txt", name2 = "output.txt"; // Файла ввода/вывода информации

index = 0.0;

InputMas(x, y, name1); // Считывание входных данных из заданного файла

if(Diagonal\_dominance(x) == 0) // Проверка на диагональное преобладание

{

Assignment(x1, x);

Test(x1, y);

Jacobi\_method(x, y);

if ((NormC\_of\_the\_matrix(x) < 1.0)) // Проверка матрицы на сильную норму

{

Assignment(a, y);

double temp[N];

do // Вычисление методом простых итераций

{

MultiMas(x, a, temp);

for (int j=0;j<N;j++)

{

b[j] = a[j];

a[j] = y[j] + temp[j];

}

NullMas(temp);

index++;

}

while (!Change\_accuracy(x, a, b, accuracy)); // Проверка выполнения условия окончания

Test(x, y);

MultiMas(x1, a, temp1); // Проверка правильности найденного решения

OutputAnswer(a, name2, index); // Вывод результата в файл

}

else

qDebug("Error: Norm of the matrix !");

}

else

qDebug("Error: Diagonal dominance !");

return asd.exec();

}

**3.2.**wid.h

#ifndef WID\_H

#define WID\_H

#include <stdio.h>

#include <cmath>

#include <fstream>

#include <iostream>

#include <stdlib.h>

#include <QFile>

#include <QString>

#define N 4

#define minimum 0.00000001

using namespace std;

void Test(double \_mas[N][N], double y[N]); /// Вывод матрицы и вектора правой части в консоль

void SwapRows(double \_mas[N][N], int \_i, int \_j); /// Поменять строки в данной матрице местами

void Assignment(double \_mas[N][N], double \_mas2[N][N]); /// Функция присваивания одного двухмнерного массива другому

void Assignment(double \_mas[N], double \_mas2[N]); /// Функция присваивания одного одномерного массива другому

void Transp(double \_mas[N][N]); /// Транспонирование матрицы

void Determinant(double \_mas[N][N]); /// Вычисление определителя

void IdentityMas(double \_mas[N][N]); /// Создание единичной матрицы

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N]); /// Произведение двухмерных массивов

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N]); /// Произведение одномерных массивов

void Jacobi\_method(double \_x[N][N], double \_y[N]); /// Метод Якоби

double NormC\_of\_the\_matrix(double \_mas[N][N]); /// Вычисление сильной нормы матрицы

double NormC\_of\_the\_matrix(double \_mas[N]); /// Вычисление сильной нормы вектора

int Diagonal\_dominance(double \_mas[N][N]); /// Проверка на диагональное преобладание матрицы

bool Change\_accuracy(double \_x[N][N], double \_y1[N], double \_y2[N], double accuracy); /// Проверка удовлетворения заданной точности

void InputMas(double x[N][N], double y[N], QString name1); /// Считывание входных данных из указанного файла

void OutputAnswer(double a[N], QString name2, int index); /// Вывод результата в указанный файл

void NullMas(double \_mas[N][N]); /// Обнуление двухмерного массива

void NullMas(double \_mas[N]); /// Обнуление одномерного массива

#endif // WID\_H

**3.3.**wid.cpp

#include "wid.h"

void InputMas(double x[N][N], double y[N], QString name1)

{

ifstream in;

in.open(QFile::encodeName(name1).data());

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

{

for(int j=0;j<N;j++)

in >> x[i][j];

in >> y[i];

}

}

void OutputAnswer(double a[N], QString name2, int index)

{

ofstream out;

out.open(QFile::encodeName(name2).data());

out << "Answer\n";

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

{

if(abs(a[i]) < minimum)

out << "x" << i+1 << " = 0\n";

else

out << "x" << i+1 << " = " << a[i] << "\n";

}

out << "\nNumber of itteration: " << index << "\n";

out.close();

}

void Test(double \_mas[N][N], double y[N])

{

printf("\tTest\n");

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

{

for(int j=0;j<N;j++)

{

if(abs(\_mas[i][j]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i][j]);

}

printf("\t%.2f\n",y[i]);

}

printf("\n");

}

bool Change\_accuracy(double \_x[N][N], double \_y1[N], double \_y2[N], double accuracy)

{

double temp[N];

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

{

temp[i] = \_y1[i] - \_y2[i];

}

double a = NormC\_of\_the\_matrix(temp);

double b = NormC\_of\_the\_matrix(\_x);

if ((b/(1-b))\*a >= accuracy)

return false;

else

return true;

}

int Diagonal\_dominance(double \_mas[N][N])

{

double a[N];

NullMas(a);

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

{

for (int j = 0;j<N;j++)

{

if(j!=i)

{

a[i] += abs(\_mas[j][i]);

}

}

}

for (int k = 0; k<N; k++)

{

// qDebug("%f\t%f",abs(\_mas[k][k]),a[k]);

if(abs(\_mas[k][k]) <= a[k])

return 1;

}

return 0;

}

double NormC\_of\_the\_matrix(double \_mas[N])

{

double r = abs(\_mas[0]);

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

{

if (r < abs(\_mas[i]))

{

r = abs(\_mas[i]);

}

}

return r;

}

double NormC\_of\_the\_matrix(double \_mas[N][N])

{

double a[N];

NullMas(a);

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

{

for (int j = 0;j<N;j++)

{

a[i] += abs(\_mas[i][j]);

}

}

double r = a[0];

for (int k = 1; k<N; k++)

{

if (r < a[k])

r = a[k];

}

return r;

}

void Jacobi\_method(double \_x[N][N], double \_y[N])

{

double a[N][N];

double b[N];

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

{

if ( (\_x[i][i] == 0.0)&&(i!=N-1))

SwapRows(\_x, i, i+1);

b[i] = \_y[i] / \_x[i][i];

for(int j=0;j<N;j++)

{

if(j!=i)

a[i][j] = - (\_x[i][j] / \_x[i][i]);

else

a[i][j] = 0.0;

}

}

Assignment(\_x, a);

Assignment(\_y, b);

}

void Transp(double \_mas[N][N])

{

double temp;

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

{

for(int j=i; j<N; j++)

{

temp = \_mas[i][j];

\_mas[i][j] = \_mas[j][i];

\_mas[j][i] = temp;

}

}

}

void SwapRows(double \_mas[N][N], int \_i, int \_j)

{

double temp;

for(int k=0;k<N;k++)

{

temp = \_mas[\_i][k];

\_mas[\_i][k] = \_mas[\_j][k];

\_mas[\_j][k] = temp;

}

}

void Assignment(double \_mas[N][N], double \_mas2[N][N])

{

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

{

for(int j=0;j<N;j++)

\_mas[i][j] = \_mas2[i][j];

}

}

void Assignment(double \_mas[N], double \_mas2[N])

{

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

{

\_mas[i] = \_mas2[i];

}

}

void Determinant(double \_mas[N][N])

{

double a = 1.0;

for(int j=0;j<N+1;j++)

{

a \*= (-1.0);

}

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

a \*= \_mas[i][i];

}

void NullMas(double \_mas[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas[i][j] = 0.0;

}

}

}

void NullMas(double \_mas[N])

{

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

{

\_mas[i] = 0.0;

}

}

void IdentityMas(double \_mas[N][N])

{

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

{

\_mas[i][i] = 1.0;

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N])

{

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

{

for(int j=0;j<N;j++)

{

for(int k=0;k<N;k++)

\_mas3[i][j] += (\_mas[i][k])\*(\_mas2[k][j]);

}

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N])

{

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

{

for(int k=0;k<N;k++)

\_mas3[i] += (\_mas[i][k])\*(\_mas2[k]);

}

}

**3.4.** Входные данные(файл input3.txt)

20 5 7 1 -117

-1 13 0 -7 -1

4 -6 17 5 49

-9 8 4 -25 -21

**3.5.** Выходные данные

Error: Diagonal dominance!

Здесь не выполнено условие диагонального преобладания.

**4.Метод Зейделя**

**4.1.**mai.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication asd(argc, argv);

double y[N], a[N], b[N], temp1[N], x[N][N], x1[N][N];

int index; // Колличество итераций

double accuracy = 0.01; // Заданная точность вычислений

QString name1 = "input.txt", name2 = "output.txt";

InputMas(x, y, name1);

index = 0.0;

NullMas(a);

if(Diagonal\_dominance(x) == 0)// Проверка на диагональное преобладание исходной матрицы

{

Assignment(x1, x);

//Test(x1, y);

do

{

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

{

if ( (x[i][i] == 0.0)&&(i!=N-1))

SwapRows(x, i, i+1);

double var = 0;

for (int j = 0; j < N; j++)

if (j != i) var += (x[i][j] \* a[j]);

b[i] = a[i];

a[i] = (y[i] - var) / x[i][i];

}

index++;

}

while (!Converge(a, b, accuracy)); // Проверка удовлетворения заданной точности

cout << "\tMatrix:\n";

Test(x, y);

index--;

OutputAnswer(a, index, name2); // Вывод результатов в фаил

MultiMas(x1, a, temp1);//Проверяем,что произведение исходной

//матрицы на вектор корней равен правой части

cout << "\tAnswer:\n";

Test(a);// Вывод результата на консоль

}

else

qDebug("Error : Diagonal dominance !");

return asd.exec();

}

**4.1.**wid.h

#ifndef WID\_H

#define WID\_H

#include <iostream>

#include <QByteArray>

#include <stdio.h>

#include <cmath>

#include <fstream>

#include <iostream>

#include<QFile>

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.0000000000000001

#define N 4

using namespace std;

void Test(double \_mas[N]);/// Вывод на консоль одномерного массива

void Test(double \_mas[N][N], double y[N]);/// Вывод на консоль матрицы со столбом правой части системы

bool Converge(double \_a[N], double \_b[N], double accuracy);/// Проверка выполнения заданной точности

void SwapRows(double \_mas[N][N], int \_i, int \_j);/// Меняет местами строки в двумерном массиве

void Assignment(double \_mas[N][N], double \_mas2[N][N]);/// Поэлементное присвание двумерных массивов

void Assignment(double \_mas[N], double \_mas2[N]);/// Поэлементное присванивание одномерных массивов

void NullMas(double \_mas[N][N]);/// Обнуление двумерного массива

void NullMas(double \_mas[N]);/// Обнуление одномерного массива

void IdentityMas(double \_mas[N][N]);/// Создание еденичной матрицы

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N]);/// Перемножение матриц

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N]);/// Перемножение матрицы на вектор столбец

double NormC\_of\_the\_matrix(double \_mas[N][N]);/// Нахождение сильной нормы матрицы

double NormC\_of\_the\_matrix(double \_mas[N]);/// Нахождение сильной нормы вектора

int Diagonal\_dominance(double \_mas[N][N]);/// Проверка на диагональное преобладание

void InputMas(double x[N][N], double y[N], QString name1);/// Считывание данных из файла

void OutputAnswer(double a[N], int index, QString name2);/// Вывод данных(ответов) в файл

#endif // WID\_H

**4.1.**wid.cpp

#include "wid.h"

void InputMas(double x[N][N], double y[N], QString name1)

{

ifstream in;

in.open(QFile::encodeName(name1).data());

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

{

for(int j=0;j<N;j++)

in >> x[i][j];

in >> y[i];

}

}

void OutputAnswer(double a[N], int index, QString name2)

{

ofstream out;

out.open(QFile::encodeName(name2).data());

out << "Answer\n";

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

{

if (abs(a[i]) < minimum)

out << "x" << i+1 << " = 0\n";

else

out << "x" << i+1 << " = " << a[i] << "\n";

}

out << "\nNumber of itteration: " << index << "\n";

out.close();

}

void Test(double \_mas[N])

{

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

{

printf("%.2f\t",\_mas[i]);

}

}

void Test(double \_mas[N][N], double y[N])

{

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

{

for(int j=0;j<N;j++)

{

if(abs(\_mas[i][j]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i][j]);

}

printf("\t%.2f\n",y[i]);

}

printf("\n");

}

bool Converge(double \_a[N], double \_b[N], double accuracy)

{

double temp[N];

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

{

temp[i] = \_a[i] - \_b[i];

}

double a = NormC\_of\_the\_matrix(temp);

if (a >= accuracy)

return false;

return true;

}

int Diagonal\_dominance(double \_mas[N][N])

{

double a[N];

NullMas(a);

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

{

for (int j = 0;j<N;j++)

{

if(j!=i)

{

a[i] += abs(\_mas[j][i]);

}

}

}

for (int k = 0; k<N; k++)

{

if(abs(\_mas[k][k]) <= a[k])

return 1;

}

return 0;

}

double NormC\_of\_the\_matrix(double \_mas[N])

{

double r = abs(\_mas[0]);

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

{

if (r < abs(\_mas[i]))

{

r = abs(\_mas[i]);

}

}

return r;

}

double NormC\_of\_the\_matrix(double \_mas[N][N])

{

double a[N];

NullMas(a);

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

{

for (int j = 0;j<N;j++)

{

a[i] += abs(\_mas[i][j]);

}

}

double r = a[0];

for (int k = 1; k<N; k++)

{

if (r < a[k])

r = a[k];

}

return r;

}

void SwapRows(double \_mas[N][N], int \_i, int \_j)

{

double temp;

for(int k=0;k<N;k++)

{

temp = \_mas[\_i][k];

\_mas[\_i][k] = \_mas[\_j][k];

\_mas[\_j][k] = temp;

}

}

void Assignment(double \_mas[N][N], double \_mas2[N][N])

{

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

{

for(int j=0;j<N;j++)

\_mas[i][j] = \_mas2[i][j];

}

}

void Assignment(double \_mas[N], double \_mas2[N])

{

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

{

\_mas[i] = \_mas2[i];

}

}

void NullMas(double \_mas[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas[i][j] = 0.0;

}

}

}

void NullMas(double \_mas[N])

{

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

{

\_mas[i] = 0.0;

}

}

void IdentityMas(double \_mas[N][N])

{

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

{

\_mas[i][i] = 1.0;

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N])

{

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

{

for(int j=0;j<N;j++)

{

for(int k=0;k<N;k++)

\_mas3[i][j] += (\_mas[i][k])\*(\_mas2[k][j]);

}

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N])

{

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

{

for(int k=0;k<N;k++)

\_mas3[i] += (\_mas[i][k])\*(\_mas2[k]);

}

}

**4.4.** Входные данные(файл input3.txt)

20 5 7 1 -117

-1 13 0 -7 -1

4 -6 17 5 49

-9 8 4 -25 -21

**4.5.** Выходные данные

Error: Diagonal dominance!

Здесь не выполнено условие диагонального преобладания.

**5.Метод Вращений.**

**5.1.**mai.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

double x[N][N], y[N], u[N][N], temp[N][N], temp2[N][N], temp3[N][N];

int index; // Счётчик колличества итераций

double accuracy = 0.0001; // Заданная точность

QString name1 = "input.txt", name2 = "output.txt"; // Файлы считывания и записи информации

InputMas(x, name1);// Считывание из файла

IdentityMas(temp2);

index = 0;

if(Symmetry(x) == 0 ) // Проверка на симметричность исходной матрицы

{

do // Метод вращений

{

NullMas(temp);

NullMas(u);

IdentityMas(u);

int index\_i,index\_j;

double fi;

MaxElement(x,index\_i,index\_j);

qDebug("Max[%d][%d] = %.2f", index\_i,index\_j, x[index\_i][index\_j]);

if(x[index\_i][index\_i]!=x[index\_j][index\_j])

fi = 0.5 \* qAtan((2 \* x[index\_i][index\_j]) / (x[index\_i][index\_i] - x[index\_j][index\_j]));

else

fi = pi/4;

qDebug("Fi = %f", fi);

ChangeU(u, index\_i, index\_j, fi);

Test(u);

Transp(u);

MultiMas(u, x, temp);

Transp(u);

MultiMas(temp, u, x);

Test(x);

MultiMas(temp2, u, temp3);

Assignment(temp2, temp3);

index++;

}

while(!CriterionTermination(x, accuracy));// Проверка выполнения условия окончания

Eigenvalues(x, y);// Выделение собственных значений из матрицы

index--;

Proverka(temp2);

OutputAnswer(x, temp2, y, index, name2);// Вывод в фаил

}

else

qDebug("Matrix is not Symmetry !");

return a.exec();

}

**5.2.**wid.h

#ifndef WID\_H

#define WID\_H

#include <QVector>

#include <QObject>

#include <qmath.h>

#include <stdio.h>

#include <cmath>

#include <fstream>

#include <QtGui/QFileDialog>

#include <QDebug>

#include <QtDebug>

#include <iostream>

#include <QByteArray>

using namespace std;

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.00001

#define N 3 //Размерность исходной матрицы

void Test(double \_mas[N][N]);/// Вывод на консоль двумерного массива

void Test(double \_mas[N]);/// Вывод на консоль одномерного массива

void SwapRows(double \_mas[N][N], int \_i, int \_j);/// Меняет местами строки в двумерном массиве

void SwapRows(double \_mas[N], int \_i, int \_j);/// Меняет местами элементы в одномерном массиве

void Assignment(double \_mas[N][N], double \_mas2[N][N]);/// Поэлементное присванивание двумерных массивов

void Assignment(double \_mas[N], double \_mas2[N]);/// Поэлементное присванивание одномерных массивов

void Transp(double \_mas[N][N]);/// Транспонирование матриц

void NullMas(double \_mas[N][N]);/// Обнуление двумерного массива

void NullMas(double \_mas[N]);/// Обнуление одномерного массива

void IdentityMas(double \_mas[N][N]);/// Создание еденичной матрицы

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N]);/// Перемножение матриц

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N]);/// Перемножение матрицы на вектор столбец

void MaxElement(double \_mas[N][N], int &\_i, int &\_j);/// Поиск максимального элемента

void ChangeU(double \_mas[N][N], int \_i, int \_j, double \_fi); /// Создание матрицы вращений

bool CriterionTermination(double \_mas[N][N], double accuracy);/// Проверка удовлетворения заданному критерию точности

int Symmetry(double \_mas[N][N]);/// Проверка на симметричность исходной матрицы

void Eigenvalues(double \_mas[N][N], double \_mas2[N]);/// Выделение собственных значений из матрицы

void Proverka(double \_mas[N][N]);/// Проверка на ортогональность собственных векторов

void InputMas(double x[N][N], QString name1); /// Считывание из файла

void OutputAnswer(double x[N][N], double temp2[N][N], double y[N], int index, QString name2);/// Вывод в фаил

#endif // WID\_H

**5.3.**wid.cpp

#include "wid.h"

void InputMas(double x[N][N], QString name1)

{

ifstream in;

in.open(QFile::encodeName(name1).data());

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

{

for(int j=0;j<N;j++)

in >> x[i][j];

}

in.close();

}

void OutputAnswer(double x[N][N], double temp2[N][N], double y[N], int index, QString name2)

{

ofstream out;

out.open(QFile::encodeName(name2).data());

out << "Matrix Lambda\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(x[i][j]) < minimum)

out << "0\t";

else

out << x[i][j] << "\t";

}

out << "\n";

}

out << "\nMatrix Own Vectors\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(temp2[i][j]) < minimum)

out << "0\t";

else

out << temp2[i][j] << "\t";

}

out << "\n";

}

out << "\nEigenvalues\n";

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

{

if (abs(y[i]) < minimum)

out << "lambda" << i+1 << " = 0\n";

else

out << "lambda" << i+1 << " = " << y[i] << "\n";

}

out << "\nNumber of itteration: " << index;

out.close();

}

void Test(double \_mas[N][N])

{

printf("\tTest\n");

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

{

for(int j=0;j<N;j++)

{

if(abs(\_mas[i][j]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i][j]);

}

printf("\n");

}

printf("\n");

}

void Test(double \_mas[N])

{

printf("\tTest\n");

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

{

if(abs(\_mas[i]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i]);

}

printf("\n\n");

}

void Proverka(double \_mas[N][N])

{

double \_mas2[N];

NullMas(\_mas2);

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

{

for (int j=0;j<N;j++)

\_mas2[i] += \_mas[j][i]\*\_mas[j][i+1];

}

Test(\_mas2);

}

void Eigenvalues(double \_mas[N][N], double \_mas2[N])

{

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

\_mas2[i]=\_mas[i][i];

}

int Symmetry(double \_mas[N][N])

{

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

{

for(int j=0;j<N;j++)

{

if(\_mas[i][j] != \_mas[j][i])

return 1;

}

}

return 0;

}

bool CriterionTermination(double \_mas[N][N], double accuracy)

{

double a = 0.0;

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

{

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

a += (\_mas[i][j])\*(\_mas[i][j]);

}

qDebug("a = %f\t;\t\taccuracy = %f",qSqrt(a), accuracy);

if (qSqrt(a) < accuracy)

return true;

return false;

}

void ChangeU(double \_mas[N][N], int \_i, int \_j, double \_fi)

{

\_mas[\_i][\_j] = -qSin(\_fi);

\_mas[\_j][\_i] = qSin(\_fi);

\_mas[\_i][\_i] = qCos(\_fi);

\_mas[\_j][\_j] = qCos(\_fi);

}

void MaxElement(double \_mas[N][N], int &\_i,int &\_j)

{

\_i = 0; \_j = 1;

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

{

for(int j=0;(j<N);j++)

{

if((abs(\_mas[\_i][\_j])<abs(\_mas[i][j]))&&(i!=j))

{

\_i = i;

\_j = j;

}

}

}

}

void Transp(double \_mas[N][N])

{

double temp;

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

{

for(int j=i; j<N; j++)

{

temp = \_mas[i][j];

\_mas[i][j] = \_mas[j][i];

\_mas[j][i] = temp;

}

}

}

void SwapRows(double \_mas[N], int \_i, int \_j)

{

double temp;

temp = \_mas[\_i];

\_mas[\_i] = \_mas[\_j];

\_mas[\_j] = temp;

}

void SwapRows(double \_mas[N][N], int \_i, int \_j)

{

double temp;

for(int k=0;k<N;k++)

{

temp = \_mas[\_i][k];

\_mas[\_i][k] = \_mas[\_j][k];

\_mas[\_j][k] = temp;

}

}

void Assignment(double \_mas[N][N], double \_mas2[N][N])

{

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

{

for(int j=0;j<N;j++)

\_mas[i][j] = \_mas2[i][j];

}

}

void Assignment(double \_mas[N], double \_mas2[N])

{

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

{

\_mas[i] = \_mas2[i];

}

}

void NullMas(double \_mas[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas[i][j] = 0.0;

}

}

}

void NullMas(double \_mas[N])

{

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

{

\_mas[i] = 0.0;

}

}

void IdentityMas(double \_mas[N][N])

{

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

{

\_mas[i][i] = 1.0;

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas3[i][j] = 0;

for(int k=0;k<N;k++)

\_mas3[i][j] += (\_mas[i][k])\*(\_mas2[k][j]);

}

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N])

{

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

{

for(int k=0;k<N;k++)

\_mas3[i] += (\_mas[i][k])\*(\_mas2[k]);

}

}

**5.4.**Входные данные (файл input2.txt)

**0 -7 7**

**-7 -9 -5**

**7 -5 -1**

**5.4.**Выходные данные (файл otput.txt)

Matrix Lambda

10.3014 1.2961e-005 0

1.2961e-005 -12.9621 0

0 0 -7.3393

Matrix Own Vectors(матрица собственных векторов)

0.686013 0.413773 -0.59848

-0.405322 0.900429 0.157928

0.604235 0.134236 0.785417

Eigenvalues (собственные значения)

lambda1 = 10.3014

lambda2 = -12.9621

lambda3 = -7.3393

Number of itteration: 5

**6. QR алгоритм**

**6.1.**main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

double x[N][N], y[N], q[N][N], temp[N][N], temp2[N][N], re[N], im[N];

int index; // Счётчик итераций

double accuracy = 0.0001; // Заданная точность вычислений

QString name1 = "input3.txt", name2 = "output.txt"; // Файлы считывания и записи информации

IdentityMas(q);

InputMas(x, name1); // Считывание из файла

double xn[N][N], qn[N][N], ren[N], imn[N];

index = 0;

do //QR-разложение

{

Assignment(ren, re);

Assignment(imn, im);

Assignment(xn, x);

QR(xn, qn);

Proverka(xn, qn, temp, temp2);

MultiMas(xn, qn, x);

Assignment(temp, q);

MultiMas(temp, qn, q);

Complex(x, re, im);

Test(re);

Test(im);

index++;

}

while(!CriterionTermination(x, re, im, ren, imn, accuracy)); // Проверка выполнения условия окончания

Eigenvalues(x, y); // Выделение собственных значений из матрицы

Proverka2(q);

OutputAnswer(x, q, y, re, im, index, name2); // Вывод в фаил

return a.exec();

}

**6.2.** wid.h

#ifndef WID\_H

#define WID\_H

#include <QFile>

#include <QVector>

#include <QObject>

#include <qmath.h>

#include <QDebug>

#include <QtDebug>

#include <iostream>

#include <QByteArray>

#include <stdio.h>

#include <cmath>

#include <fstream>

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.000000000000001

#define N 3 //Размерность исходной матрицы

using namespace std;

void Test(double \_mas[N][N]);/// Вывод на консоль одномерного массива

void Test(double \_mas[N]);/// Вывод на консоль одномерного массива

void SwapRows(double \_mas[N][N], int \_i, int \_j);/// Меняет местами строки в двумерном массиве

void SwapRows(double \_mas[N], int \_i, int \_j);/// Меняет местами элементы в одномерном массиве

void Assignment(double \_mas[N][N], double \_mas2[N][N]);/// Поэлементное присванивание двумерных массивов

void Assignment(double \_mas[N], double \_mas2[N]);/// Поэлементное присванивание одномерных массивов

void Transp(double \_mas[N][N]);/// Транспонирование матриц

void NullMas(double \_mas[N][N]);/// Обнуление двумерного массива

void NullMas(double \_mas[N]);/// Обнуление одномерного массива

void IdentityMas(double \_mas[N][N]);/// Создание еденичной матрицы

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N]);/// Перемножение матриц

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N]);/// Перемножение матрицы на вектор столбец

void MaxElement(double \_mas[N][N], int &\_i, int &\_j);/// Поиск максимального элемента

void ChangeH(double \_mas[N], double \_mas2[N][N]); /// Создание матрицы H

bool CriterionTermination(double \_mas[N][N], double \_re[N], double \_im[N], double \_ren[N], double \_imn[N], double accuracy);/// Проверка удовлетворения заданному критерию точности

int Symmetry(double \_mas[N][N]); /// Проверка на симметричность исходной матрицы

void Eigenvalues(double \_mas[N][N], double \_mas2[N]); /// Выделение собственных значений из матрицы

void Proverka(double \_x[N][N],double \_q[N][N], double temp[N][N], double temp2[N][N]); /// Проверка на правильность QR-разложения и на ортогональность матрицы Q

void Proverka2(double \_mas[N][N]); /// Проверка на ортогональность собственных векторов

void QR(double \_x[N][N], double \_q[N][N]);/// QR - разложение

double Sign(double \_a); /// Определение знака числа

double NormVector(double \_mas[N][N], int \_j); /// Нормировка вектора

double MultiVector1(double \_mas[N]); /// Перемножение транспонированнного ветора на на исходный

void MultiVector2(double \_mas[N], double \_mas2[N][N]); /// Перемножение исходного вектора на транспонированный

double Discriminant(double \_mas[N][N], int \_i); /// Вычисление определителя

void Complex(double \_x[N][N], double \_re[N], double \_im[N]); /// Вычисление комплексных собственных значений

void InputMas(double x[N][N], QString name1); /// Считывание из файла

void OutputAnswer(double x[N][N], double q[N][N], double y[N], double re[N], double im[N], int index, QString name2); /// Вывод в фаил

#endif // WID\_H

**6.3.** wid.cpp

#include "wid.h"

void InputMas(double x[N][N], QString name1)

{

ifstream in;

in.open(QFile::encodeName(name1).data());

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

{

for(int j=0;j<N;j++)

in >> x[i][j];

}

in.close();

}

void OutputAnswer(double x[N][N], double q[N][N], double y[N], double re[N], double im[N], int index, QString name2)

{

ofstream out;

out.open(QFile::encodeName(name2).data());

out << "Matrix R \* Q\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(x[i][j]) < minimum)

out << "0\t";

else

out << x[i][j] << "\t";

}

out << "\n";

}

out << "\nMatrix Own Vectors\n";

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

{

for(int j=0;j<N;j++)

{

if(abs(q[i][j]) < minimum)

out << "0\t";

else

out << q[i][j] << "\t";

}

out << "\n";

}

out << "\nEigenvalues\n";

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

{

if((abs(re[i]) < minimum)&&(abs(im[i]) < minimum))

{

if(abs(y[i]) < minimum)

out << "lambda" << i+1 << " = 0\n";

else

out << "lambda" << i+1 << " = " << y[i] << "\n";

}

else

{

if(im[i] < 0.0)

out << "lambda" << i+1 << " = " << re[i] << " - " << abs(im[i]) << "\*i\n";

else

out << "lambda" << i+1 << " = " << re[i] << " + " << abs(im[i]) << "\*i\n";

}

}

out << "\nNumber of itteration: " << index;

out.close();

}

void Test(double \_mas[N][N])

{

printf("\tTest\n");

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

{

for(int j=0;j<N;j++)

{

if(abs(\_mas[i][j]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i][j]);

}

printf("\n");

}

printf("\n");

}

void Test(double \_mas[N])

{

printf("\tTest\n");

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

{

if(abs(\_mas[i]) < minimum)

printf("0.00\t");

else

printf("%.2f\t",\_mas[i]);

}

printf("\n\n");

}

void QR(double \_x[N][N], double \_q[N][N])

{

double v[N], xn[N][N], h[N][N], hn[N][N];

NullMas(\_q);

IdentityMas(\_q);

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

{

NullMas(v);

NullMas(h);

Assignment(xn, \_x);

Assignment(h, \_q);

for(int j=0;j<N;j++)

{

if(j == i)

v[j] = xn[j][i] + Sign(xn[j][i])\*NormVector(xn, i);

else if(j < i)

v[j] = 0.0;

else

v[j] = xn[j][i];

}

ChangeH(v, hn);

MultiMas(hn, xn, \_x);

MultiMas(h, hn, \_q);

}

}

void ChangeH(double \_mas[N],double \_mas2[N][N])

{

double vtv, e[N][N], vvt[N][N];

NullMas(e);

IdentityMas(e);

vtv = MultiVector1(\_mas);

MultiVector2(\_mas, vvt);

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

{

for (int j=0; j<N; j++)

\_mas2[i][j] = e[i][j] - 2.0\*vvt[i][j]/vtv;

}

}

void MultiVector2(double \_mas[N], double \_mas2[N][N])

{

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

{

for (int j=0; j<N; j++)

\_mas2[i][j] = \_mas[i] \* \_mas[j];

}

}

double MultiVector1(double \_mas[N])

{

double a = 0.0;

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

a += \_mas[i] \* \_mas[i];

return a;

}

double NormVector(double \_mas[N][N], int \_j)

{

double a = 0.0;

for (int i=\_j; i<N; i++)

{

a += \_mas[i][\_j] \* \_mas[i][\_j];

}

return qSqrt(a);

}

double Sign(double \_a)

{

if (\_a < 0)

return -1.0;

else if (\_a == 0)

return 0.0;

else

return 1.0;

}

void Proverka(double \_x[N][N],double \_q[N][N], double temp[N][N], double temp2[N][N])

{

MultiMas(\_q, \_x, temp);

Test(temp);

Assignment(temp, \_q);

Transp(temp);

MultiMas(\_q, temp, temp2);

Test(temp2);

}

void Proverka2(double \_mas[N][N])

{

double \_mas2[N];

NullMas(\_mas2);

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

{

for (int j=0;j<N;j++)

\_mas2[i] += \_mas[j][i]\*\_mas[j][i+1];

}

Test(\_mas2);

}

void Eigenvalues(double \_mas[N][N], double \_mas2[N])

{

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

{

if(Discriminant(\_mas, i) > 0.0)

{

\_mas2[i] = \_mas[i][i];

if(i==N-2)

\_mas2[i+1] = \_mas[i+1][i+1];

}

}

}

void Complex(double \_x[N][N], double \_re[N], double \_im[N])

{

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

{

if(Discriminant(\_x, i) < 0.0)

{

\_re[i] = 0.5 \* (\_x[i][i] + \_x[i+1][i+1]);

\_re[i+1] = 0.5 \* (\_x[i][i] + \_x[i+1][i+1]);

\_im[i] = 0.5 \* qSqrt(0.0 - Discriminant(\_x, i));

\_im[i+1] = 0.0 - 0.5 \* qSqrt(0.0 - Discriminant(\_x, i));

}

}

}

bool CriterionTermination(double \_mas[N][N], double \_re[N], double \_im[N], double \_ren[N], double \_imn[N], double accuracy)

{

double a;

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

{

a = 0.0;

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

a += (\_mas[j][i])\*(\_mas[j][i]);

qDebug("a = %f\t;\t\taccuracy = %f",qSqrt(a), accuracy);

if (qSqrt(a) < accuracy)

return true;

qDebug("Discriminant: %f",Discriminant(\_mas, i));

if(Discriminant(\_mas, i) < 0.0)

{

qDebug("b = %f",qSqrt((\_re[i] - \_ren[i])\*(\_re[i] - \_ren[i]) + (\_im[i] - \_imn[i])\*(\_im[i] - \_imn[i])));

if(qSqrt((\_re[i] - \_ren[i])\*(\_re[i] - \_ren[i]) + (\_im[i] - \_imn[i])\*(\_im[i] - \_imn[i])) < accuracy)

return true;

}

}

qDebug("---------------");

return false;

}

double Discriminant(double \_mas[N][N], int \_i)

{

return (\_mas[\_i][\_i]+\_mas[\_i+1][\_i+1])\*(\_mas[\_i][\_i]+\_mas[\_i+1][\_i+1]) - 4 \* ((\_mas[\_i][\_i]\*\_mas[\_i+1][\_i+1]) - (\_mas[\_i][\_i+1]\*\_mas[\_i+1][\_i]));

}

void Transp(double \_mas[N][N])

{

double temp;

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

{

for(int j=i; j<N; j++)

{

temp = \_mas[i][j];

\_mas[i][j] = \_mas[j][i];

\_mas[j][i] = temp;

}

}

}

void SwapRows(double \_mas[N], int \_i, int \_j)

{

double temp;

temp = \_mas[\_i];

\_mas[\_i] = \_mas[\_j];

\_mas[\_j] = temp;

}

void SwapRows(double \_mas[N][N], int \_i, int \_j)

{

double temp;

for(int k=0;k<N;k++)

{

temp = \_mas[\_i][k];

\_mas[\_i][k] = \_mas[\_j][k];

\_mas[\_j][k] = temp;

}

}

void Assignment(double \_mas[N][N], double \_mas2[N][N])

{

NullMas(\_mas);

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

{

for(int j=0;j<N;j++)

\_mas[i][j] = \_mas2[i][j];

}

}

void Assignment(double \_mas[N], double \_mas2[N])

{

NullMas(\_mas);

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

{

\_mas[i] = \_mas2[i];

}

}

void NullMas(double \_mas[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas[i][j] = 0.0;

}

}

}

void NullMas(double \_mas[N])

{

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

{

\_mas[i] = 0.0;

}

}

void IdentityMas(double \_mas[N][N])

{

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

{

\_mas[i][i] = 1.0;

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N][N], double \_mas3[N][N])

{

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

{

for(int j=0;j<N;j++)

{

\_mas3[i][j] = 0;

for(int k=0;k<N;k++)

\_mas3[i][j] += (\_mas[i][k])\*(\_mas2[k][j]);

}

}

}

void MultiMas(double \_mas[N][N], double \_mas2[N], double \_mas3[N])

{

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

{

for(int k=0;k<N;k++)

\_mas3[i] += (\_mas[i][k])\*(\_mas2[k]);

}

}

**6.4.**Входные данные(файл input2.txt)

**5 8 -2**

**7 -2 -4**

**5 8 -1**

**6.4.**Выходные данные(файл otput.txt)

Matrix R \* Q

1.83212 -2.04653 -10.0347

-10.2763 -2.42303 5.05698

7.05684e-005 -7.40062e-005 2.5909

Matrix Own Vectors

-0.134456 0.650664 0.747368

-4.59906e+291 6.78176e+291 -6.73164e+291

-0.433683 0.639508 -0.634782

Eigenvalues

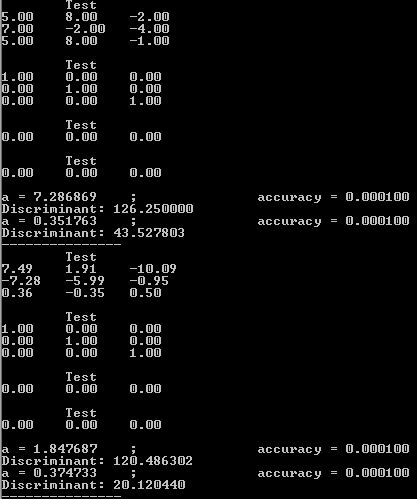
lambda1 = 1.83212

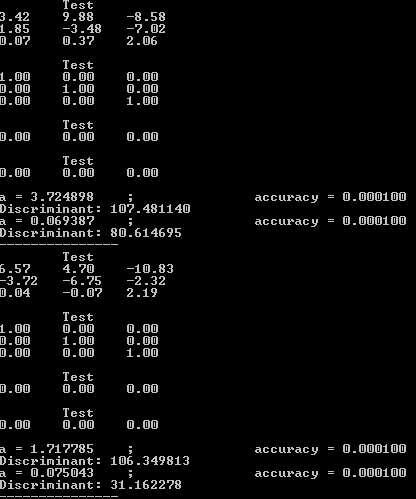
lambda2 = -2.42303

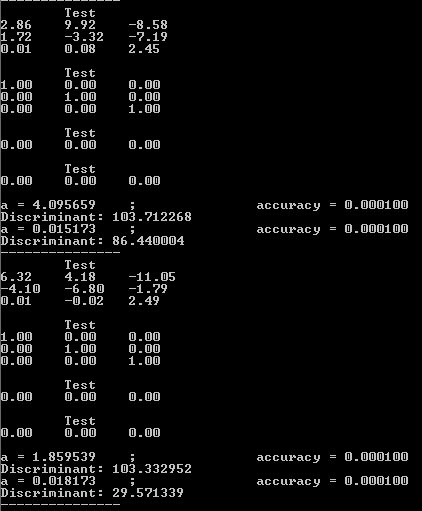
lambda3 = 2.5909

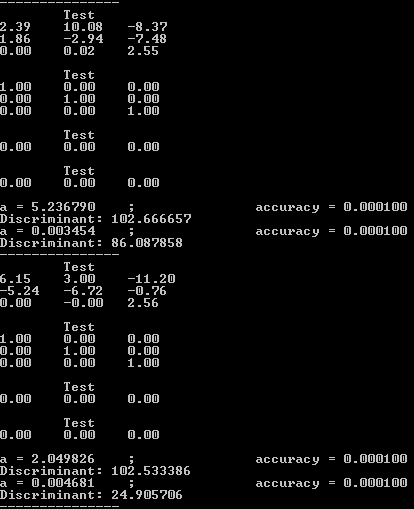
Number of itteration: 13

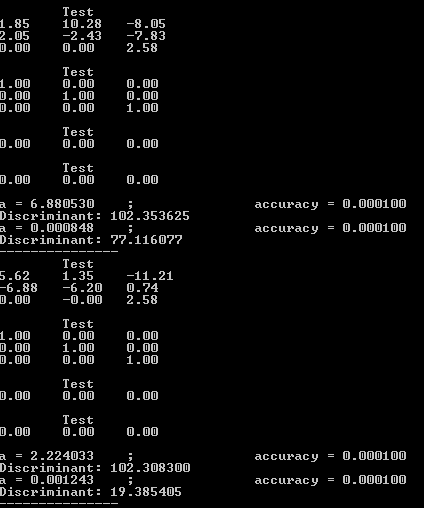
**6.6.** Результаты расчётов в консоле

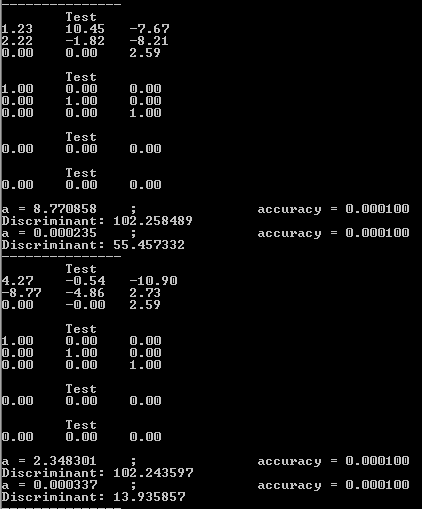
****

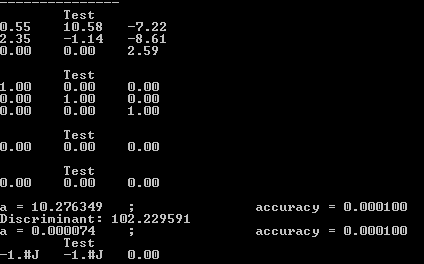
****

****

****

****

****

****

**Вывод:**

Численные методы алгебры разделяются на прямые итерационные. В каждых из методов есть свои плюсы и недостатки.

В прямых методах решение находится за конечное число шагов,что даёт гарантию для получения точного ответа. Однако такие методы, как правило очень затратны. Использовать их имеет смысл в случае матриц относительно небольших размеров. Исключение составляет метод прогонки. Метод прогонки считается эффективным методом. Однако, он применим только для трехдиагональных матриц.

В итерационных методах решение находится последовательным приближением к точному ответу,который априори не изввестен. Здесь возникают как правило три проблемы,которые формально можно озвучить так:

1. Где начать?
2. Куда идти?
3. Где закончить?

Для итерационных методов, в отличие от прямых , надо указывать точность, нужны какие-либо условия сходимости. Так ,например, достаточным условием сходимости в методе Зейделя является условие диагонального преобладания . Если оно не выполнено,то метод не сходится.