**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Теория разностных схем.

**Отчет по лабораторной работе № 3**

**Тема:** «Решение краевых задач  
для эллиптических уравнений»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа МКН-313 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Павлов Р.Е. |  |  |  |
| Принял | Ямилева А.М. |  |  |  |

**Уфа 2020**

**Цель работы:** получить навык численного решения краевых задач для уравнений эллиптического типа с использованием различных методов на примере задачи Дирихле для линейного двумерного неоднородного уравнения.

**Теоретический материал**

Рассмотрим задачу Дирихле для уравнения Пуассона:

где – единичный квадрат.

Введем равномерную сетку с шагами . Обозначим множество узлов сетки, лежащие внутри а множество узлов сетки, лежащих на . Введем сеточную функцию , принимающая значение на узле . Заменяя производные разностными аналогами, получим следующую разностную схему (разностная задача Дирихле):

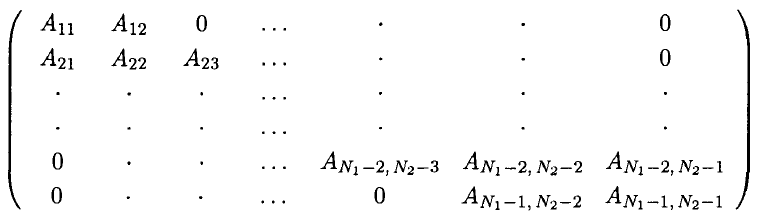
где

Подставляя (2) в (1) и перенося известные члены в правую часть, получим СЛАУ

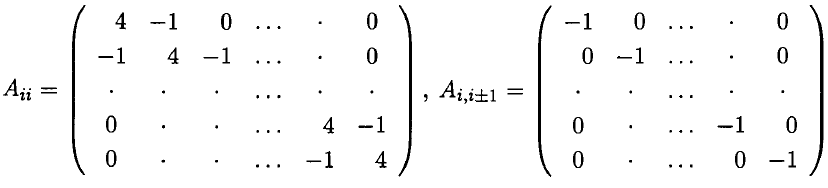
Рассмотрим случай .

Опишем структуру матрицы Упорядочим компоненты вектора неизвестных естественным образом:

Умножим обе части СЛАУ на . Тогда матрица СЛАУ будет иметь следующую блочно-трехдиагональную форму:



где матрицы размера имеют вид



**Метод простых итераций для решения разностной задачи Дирихле**

Рассмотрим простейшую явную двуслойную итерационную схему:

где

Выразим

Число выбирается из условия минимума числа итераций (такая задача решается точно):

Так как мы можем вычислить по формуле (\*), нам не нужно хранить матрицу системы

**Метод SOR для решения разностной задачи Дирихле**

Метод SOR сходится для

**Задания на лабораторную работу**

***Краевая задача для уравнения эллиптического типа***

Рассматривается задача Дирихле для линейного двумерного неоднородного эллиптического уравнения с переменными коэффициентами:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

***I. Задача Дирихле для уравнения Пуассона с постоянными коэффициентами***

Рассматривается частный случай уравнения (1) – уравнение Пуассона с постоянными коэффициентами:

(3)

По заданному в индивидуальном задании точному решению задачи (см. таблицу 1) необходимо восстановить функции и .

***Задача 1 (2 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(3) с использованием конечно-разностной схемы с шаблоном «крест» на сетке с постоянными шагами и по направлениям *x* и *y*, удовлетворяющих соотношению

Для решения получающейся СЛАУ использовать метод простых итераций. При этом матрица системы не должна храниться в памяти.

1. Исследовать зависимость погрешности решения от величины шагов сетки и построить соответствующие графики. Погрешность оценивать в равномерной норме.
2. Исследовать зависимости числа итераций от шага сетки.

***Задача 2 (2 балла).***

Решить задачу 1 с использованием для решения СЛАУ метод SOR.

Параметр релаксации либо выбирается фиксированным, либо используется формула для оптимального значения.

***Задача 3 (2 балла).***

Решить задачу 1 с использованием для решения СЛАУ любой точный метод (Гаусса, LU-разложение, матричная прогонка). В данной задаче матрицу системы можно хранить целиком в памяти, желательно только ненулевые диагонали.

Вариант 7

|  |  |  |  |
| --- | --- | --- | --- |
| №  в-та |  |  |  |
| 7 | 1 | 1 |  |

**II. Решение задачи с переменными коэффициентами**

***Задача 4 (4 балла).***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(2) с параметрами из таблиц 1 и 2 методом переменных направлений.
2. Исследовать зависимость погрешности получаемого решения от величины шага сетки, построить соответствующие графики.

Вариант 7

|  |  |  |  |
| --- | --- | --- | --- |
| №  в-та |  |  |  |
| 7 |  |  | 1 |

**Практическая реализация**

**Задача 1: Краевая задача для уравнения эллиптического типа**

Решается задача со следующими параметрами:

Подстановкой в уравнение получаем следующие функции и :

Условие остановки итерационного процесса следующее:

Построим график численного решения:

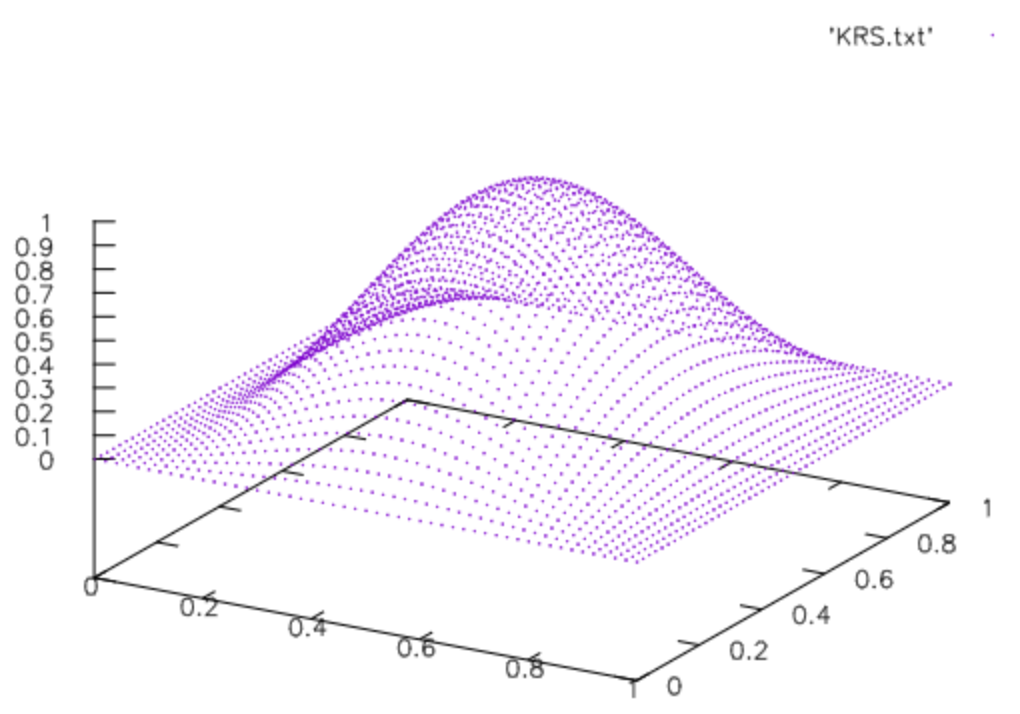


Рисунок 1. График численного решения

Построим график зависимости погрешности получаемого решения от числа узлов сетки (рисунок 2).

Рисунок 2. Зависимость погрешности от шага сетки

Построим график зависимости количества требуемых итераций от числа узлов сетки:

Рисунок 3. Зависимость количества итераций от числа узлов сетки

По полученным графикам можно сделать вывод, что с увеличением числа узлов сетки погрешность уменьшается и одновременно возрастает количество итераций, необходимое для достижения заданной точности.

**Задача 2: Решение СЛАУ методом SOR**

Параметр релаксации выбирается фиксированным и равным 1.94.

Построим график численного решения:

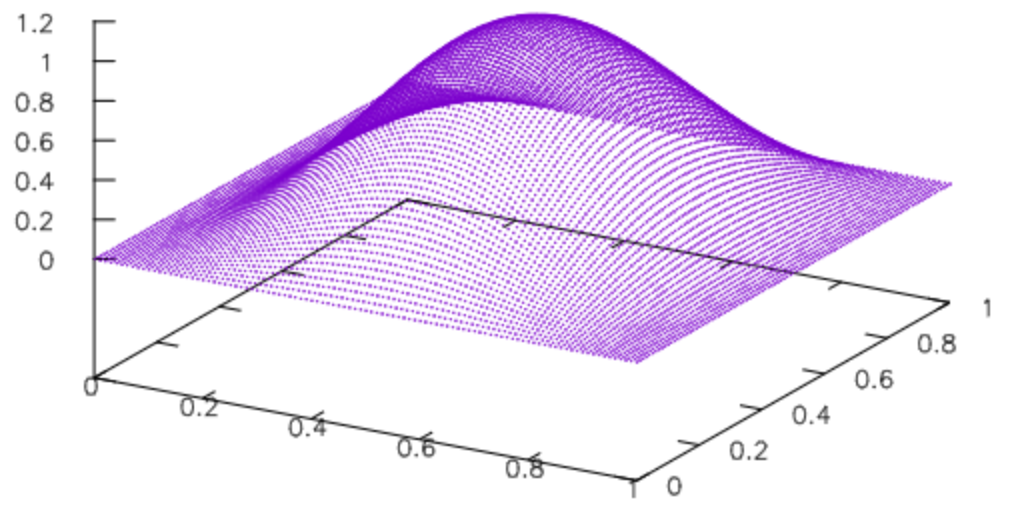


Рисунок 4. График численного решения

Построим график зависимости погрешности от числа узлов сетки (рисунок 5) и график зависимости количества итерация от числа узлов сетки (рисунок 6).

Рисунок 5. График зависимости погрешности решения от числа узлов сетки

Рисунок 6. График зависимости количества итераций от числа узлов сетки

По полученным результам можно сделать вывод, что с увеличением числа узлов сетки погрешность уменьшается и количество итераций, необхоимых для достижения заданной точности, так же уменьшается.

**Задача 3: Метод сопряженных градиентов**

Для решения задачи используется метод сопряженных градиентов.

Построим график численного решения:

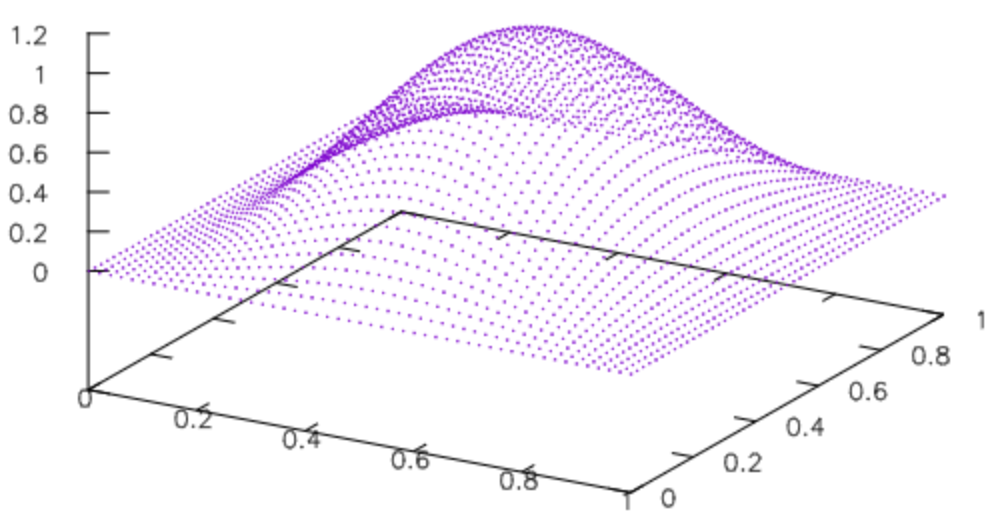


Рисунок 7. График численного решения

Построим график зависимости погрешности от числа узлов сетки и график зависимости количества итераций от числа узлов сетки.

Рисунок 8. График зависимости погрешности от числа узлов сетки

Рисунок 9. График зависимости числа итераций от числа узлов сетки.

По полученным данным можно сделать вывод, что с увеличением числа узлов сетки погрешность уменьшается, а количество итераций возрастает почти линейно.

**Вывод**

В ходе выполнения лабораторной работы был получен навык численного решения краевых задач для уравнений эллиптического типа с использованием различных методов на примере задачи Дирихле для линейного двумерного неоднородного уравнения. Было проведено сравнение различных методов решения соответствующих СЛАУ.

Для каждой поставленной задачи написана вычислительная программа на языке программирования С++, выполняющая необходимые построения и расчеты.

**Приложение**

Задачи 1.

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

#define PI 3.14159

const double epsilon = 1.0e-5;

double f(double x, double y)

{

return -PI \* PI \* sin(PI \* y) \* (5. \* cos(PI \* x) \* cos(PI \* x) - 3.);

}

double phi(double x, double y)

{

return 0;

}

double Solution(double x, double y)

{

return sin(PI \* x) \* sin(PI \* x) \* sin(PI \* y);

}

double Error(double \*y, double lx, double ly, int N)

{

double hx = lx/N;

double hy = (ly\*hx)/lx;

double max=-1.0;

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

{

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

{

if (abs(y[j\*(N+1)+i]-Solution(hx\*i,hy\*j))>=max)

max=abs(y[j\*(N+1)+i]-Solution(hx\*i,hy\*j));

}

}

return max;

}

double delta(double \*y\_0, double \*y, double N)

{

double max=-1;

double temp=0;

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

{

temp=abs(y\_0[i]-y[i]);

if (temp>max)

max=temp;

}

return max;

}

void SimpleIterations\_Puasson(double \*y, double lx, double ly, int N)

{

double hx = lx/N;

double hy = (ly\*hx)/lx;

double tau = pow(hx\*hy,2)/(2.0\*(pow(hx,2)+pow(hy,2)));

double \*y\_0 = new double[(N+1)\*(N+1)];

double D = 0;

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

{

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

{

if ((i==0)||(i==N)||(j==0)||(j==N))

{

y\_0[j\*(N+1)+i]=phi(i\*hx,j\*hy);

y[j\*(N+1)+i]=y\_0[j\*(N+1)+i];

}

else

y\_0[j\*(N+1)+i]=0;

}

}

int iter = 0;

do

{

iter++;

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

{

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

y[j\*(N+1)+i]=y\_0[j\*(N+1)+i]+tau\*((y\_0[(j+1)\*(N+1)+i]-2.0\*y\_0[j\*(N+1)+i]+y\_0[(j-1)\*(N+1)+i])/pow(hy,2)+

(y\_0[j\*(N+1)+i+1]-2.0\*y\_0[j\*(N+1)+i]+y\_0[j\*(N+1)+i-1])/pow(hx,2)+f(i\*hx,j\*hy));

}

D = delta(y\_0,y,N);

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

y\_0[i]=y[i];

}

while (D>epsilon);

cout << iter << endl;

}

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

{

int N=50;

double lx=1.0;

double ly=1.0;

if (argc == 2)

{

N = atoi(argv[1]);

}

double hx = 1. / (double)(N);

double hy = 1. / (double)(N);

double \*y = new double[(N+1)\*(N+1)];

SimpleIterations\_Puasson(y,lx,ly,N);

ofstream filewrite("KRS.txt");

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

{

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

filewrite << i\*hx << " " << j\*hy << " " << y[j\*(N+1)+i] << endl;

}

cout << Error(y,lx,ly,N) << endl;

return 0;

}

Задача 2

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

#define PI 3.14159

const double epsilon = 1.0e-10;

double f(double x, double y)

{

return -PI \* PI \* sin(PI \* y) \* (5. \* cos(PI \* x) \* cos(PI \* x) - 3.);

}

double phi(double x, double y)

{

return 0;

}

double Solution(double x, double y)

{

return sin(PI \* x) \* sin(PI \* x) \* sin(PI \* y);

}

double ChebNormErr(double \*y, double lx, double ly, int N)

{

double hx = lx/N;

double hy = (ly\*hx)/lx;

double max=-1.0;

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

{

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

{

if (abs(y[j\*(N+1)+i]-Solution(hx\*i,hy\*j))>=max)

max=abs(y[j\*(N+1)+i]-Solution(hx\*i,hy\*j));

}

}

return max;

}

double delta(double \*y\_0, double \*y, double N)

{

double max=-1;

double temp=0;

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

{

temp=abs(y\_0[i]-y[i]);

if (temp>max)

max=temp;

}

return max;

}

void SOR\_Puasson(double \*y, double lx, double ly, int N, double w)

{

double hx = lx/N;

double hy = (ly\*hx)/lx;

double \*y\_0 = new double[(N+1)\*(N+1)];

double D = 0;

int iter = 0;

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

{

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

{

if ((i==0)||(i==N)||(j==0)||(j==N))

{

y\_0[j\*(N+1)+i]=phi(i\*hx,j\*hy);

y[j\*(N+1)+i]=y\_0[j\*(N+1)+i];

}

else

y\_0[j\*(N+1)+i]=0;

}

}

do

{

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

{

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

{

y[j\*(N+1)+i]=(1-w)\*y\_0[j\*(N+1)+i]+w\*

((y\_0[j\*(N+1)+i+1]+y[j\*(N+1)+i-1])/pow(hx,2)+

(y\_0[(j+1)\*(N+1)+i]+y[(j-1)\*(N+1)+i])/pow(hy,2)+f(i\*hx,j\*hy))/(2.0/pow(hx,2)+2.0/pow(hy,2));

}

}

D = delta(y\_0,y,N);

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

y\_0[i]=y[i];

iter++;

}

while (D>epsilon);

cout << iter << endl;

}

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

{

int N=100;

double lx=1.0;

double ly=1.0;

if (argc == 2)

{

N = atoi(argv[1]);

}

double hx = 1. / (double)(N);

double hy = 1. / (double)(N);

double \*y = new double[(N+1)\*(N+1)];

SOR\_Puasson(y,lx,ly,N,1.94);

ofstream filewrite("SOR.txt");

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

{

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

filewrite << i\*hx << " " << j\*hy << " " << y[j\*(N+1)+i] << endl;

}

cout << ChebNormErr(y,lx,ly,N) << endl;

return 0;

}

Задача 3

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

#define PI 3.14159

const double epsilon = 1.0e-10;

double f(double x, double y)

{

return -PI \* PI \* sin(PI \* y) \* (5. \* cos(PI \* x) \* cos(PI \* x) - 3.);

}

inline double Acc(double \*arr, int N, int i, int j)

{

double temp=0;

int K=sqrt(N);

if ((abs(i-j)==K)&&(j>i)) { temp=arr[i\*5+4]; return temp; }

if ((abs(i-j)==K)&&(j<i)) { temp=arr[i\*5+0]; return temp; }

if (abs(i-j)<=1) { temp=arr[i\*5+j-i+2]; return temp; }

return temp;

}

double VecScalarProduct(double \*a, double \*b, int N)

{

double result=0;

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

result+=a[i]\*b[i];

return result;

}

double\* VecSum(double \*a, double \*b, double a\_coef, double b\_coef, int N)

{

double \*c=new double[N];

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

c[i]=a\_coef\*a[i]+b\_coef\*b[i];

return c;

}

double VecNorm(double \*a, int N)

{

double result=abs(a[0]);

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

{

if (abs(a[i])>result)

result=abs(a[i]);

}

return result;

}

double\* MatrixVectorProduct(double \*A, double \*v, int N)

{

double \*b = new double[N];

double temp=0;

double elemA=0;

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

{

temp=0;

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

{

elemA=Acc(A,N,i,k);

temp+=elemA\*v[k];

}

b[i]=temp;

}

return b;

}

double\* CGM(double \*A, double \*b, double \*x\_0, int N, int maxiter, double acc)

{

double \*x\_k=new double[N];

double \*x\_k1=new double[N];

double \*p\_k=new double[N];

double \*p\_k1=new double[N];

double \*r\_k=new double[N];

double \*r\_k1=new double[N];

double \*q\_k;

double alpha\_k=0, beta\_k=0;

double delta=0;

int iter=0;

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

x\_k[i]=x\_0[i];

r\_k=VecSum(b,MatrixVectorProduct(A,x\_k,N),1.0,-1.0,N);

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

p\_k[i]=r\_k[i];

do

{

iter++;

q\_k=MatrixVectorProduct(A,p\_k,N);

alpha\_k=-VecScalarProduct(r\_k,r\_k,N)/VecScalarProduct(p\_k,q\_k,N);

x\_k1=VecSum(x\_k,p\_k,1.0,-alpha\_k,N);

r\_k1=VecSum(r\_k,q\_k,1.0,alpha\_k,N);

delta=VecScalarProduct(r\_k1,r\_k1,N);

//cout << delta << endl;

if (delta<acc)

break;

beta\_k=VecScalarProduct(r\_k1,r\_k1,N)/VecScalarProduct(r\_k,r\_k,N);

p\_k1=VecSum(r\_k1,p\_k,1.0,beta\_k,N);

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

{

x\_k[i]=x\_k1[i];

r\_k[i]=r\_k1[i];

p\_k[i]=p\_k1[i];

}

}

while (iter<maxiter);

cout << iter << " Iterations" << endl;

return x\_k1;

}

double Solution(double x, double y)

{

//return (exp(x\*y)-1)\*(1-x)\*(1-y);

return sin(PI \* x) \* sin(PI \* x) \* sin(PI \* y);

}

double ChebNormErr(double \*y, double lx, double ly, int N)

{

double hx = lx/N;

double hy = (ly\*hx)/lx;

double max=-1.0;

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

{

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

{

if (abs(y[j\*(N+1)+i]-Solution(hx\*i,hy\*j))>=max)

max=abs(y[j\*(N+1)+i]-Solution(hx\*i,hy\*j));

}

}

return max;

}

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

{

int N=50;

if (argc == 2)

{

N = atoi(argv[1]);

}

double lx=1, ly=1;

double hx = lx/N;

double hy = (ly\*hx)/lx;

double \*A = new double[(N-1)\*(N-1)\*5];

double \*y\_0 = new double[(N-1)\*(N-1)];

double \*y = new double[(N+1)\*(N+1)];

for (int i=0; i<5; i++)

{

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

{

if (i==0)

{

if (j<=N-2) A[j\*5+i]=0;

else A[j\*5+i]=1/pow(hx,2);

}

if (i==1)

{

if ((j==0)||(j%(N-1)==0)) A[j\*5+i]=0;

else A[j\*5+i]=1/pow(hy,2);

}

if (i==2) A[j\*5+i]=-2\*(1/pow(hx,2)+1/pow(hy,2));

if (i==3)

{

if ((j==(N-1)\*(N-1))||((j+1)%(N-1)==0)) A[j\*5+i]=0;

else A[j\*5+i]=1/pow(hy,2);

}

if (i==4)

{

if (j>=(N-2)\*(N-1)) A[j\*5+i]=0;

else A[j\*5+i]=1/pow(hx,2);

}

}

}

double \*b = new double[(N-1)\*(N-1)];

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

{

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

{

b[j\*(N-1)+i]=-f((i+1)\*hx,(j+1)\*hy);

y\_0[j\*(N-1)+i]=0;

}

}

y\_0=CGM(A,b,y\_0,(N-1)\*(N-1),1000,epsilon);

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

{

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

{

if ((i==0)||(i==N)||(j==0)||(j==N))

y[j\*(N+1)+i]=0;

else

y[j\*(N+1)+i]=y\_0[(j-1)\*(N-1)+i-1];

}

}

ofstream filewrite("CGM.txt");

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

{

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

filewrite << i\*hx << " " << j\*hy << " " << y[j\*(N+1)+i] << endl;

}

cout << ChebNormErr(y,lx,ly,N) << endl;

return 0;

}