**Федеральное государственное бюджетное образовательное учреждение высшего 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 ПМ-353 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Шамаев И.Р. |  |  |  |
| Принял | Белевцов Н.С. |  |  |  |

**Уфа 2022**

**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.

**Теоретическая часть**

***Метод простых итераций***

Рассмотрим метод простой итерации с параметром для уравнения

Итерационный процесс в этом случае имеет вид

*h*-оптимальный выбор

***Метод Якоби***

Пусть Представим в виде , где D – матрица с диагональными элементами , а – недиагональные элементы . Тогда Таким образом, метод Якоби – модификация метода простых итераций.

Координатная форма:

***Метод SOR***

Метод релаксации - итерационный метод решения систем линейных алгебраических уравнений. Система линейных уравнений

Приводится к виду:

Выбирается начальное приближение . На каждом шаге необходимо обратиь в ноль максимальную невязку:

Ответ находится по формуле:

находится из уравнения

***Метод Гаусса с выбором ведущего элемента***

Метод Гаусса с выбором главного (максимального) элемента по столбцам является одной из модификаций метода Гаусса, позволяющих уменьшить погрешность вычислений.

Cреди элементов матрицы А выбирается наибольший по модулю называемый главным элементом. Соответственно строка с этим элементом будет главной строкой. Предположим, что , если это не так, то меняют местами первую строку со строкой p и первый столбец со столбцом q, при этом совершают перенумерацию коэффициентов и неизвестных. Теперь первая строка становится главной.

Полученное первое уравнение системы делится на и получают уравнение вида:

Где

На следующем шаге исключают, неизвестную величину из каждого уравнения исходной системы начиная со второго, путем вычитания (1), умноженного на коэффициент , при в соответствующем уравнении. Отбрасывают главную строку и первый столбец матрицы А и получают преобразованную систему уравнений

Обратный ход нахождения коэффициентов через систему (1)

**Практическая часть**

***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 необходимо восстановить функции и .

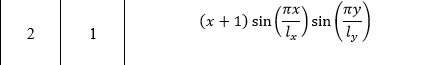
|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

***Задача 1***

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.



Решение:

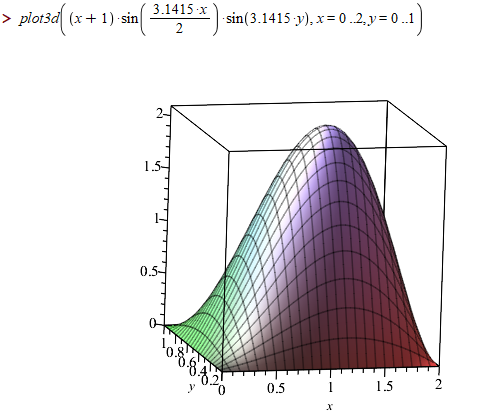


Рисунок 1. График точного решения

Условие остановки итерационного процесса следующее:

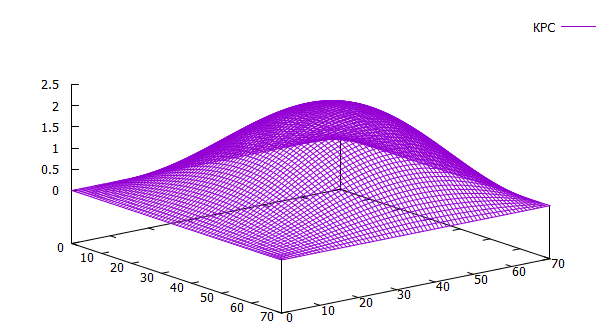


Рисунок 2. График численного решения

В таблице 1 приведены значения количества узлов, количества итераций и погрешности

|  |  |  |
| --- | --- | --- |
| **N** | **Итерации** | **Погрешность** |
| 10 | 185 | 0,0176415 |
| 30 | 1276 | 0,009577 |
| 50 | 3028 | 0,00455 |
| 70 | 5268 | 0,00440197 |
| 100 | 9307 | 0,00200728 |

Таблица 1.

Построим графики зависимости погрешности численного решения задачи и числа итераций от числа узлов сетки:

Погрешность

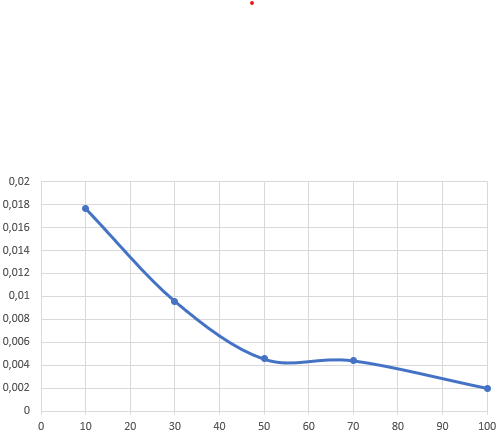
N

Рисунок 3. График зависимости погрешности от шага сетки

При увеличении числа узлов сетки погрешность уменьшается.

Кол-во итераций

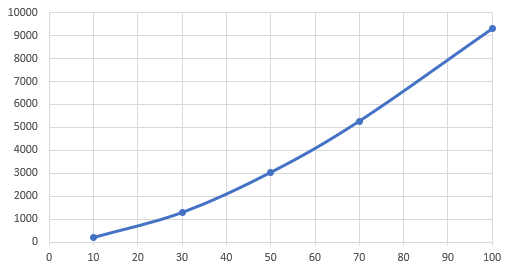
N

Рисунок 4. Зависимость числа итераций от шага сетки

С увеличением числа узлов сетки количество итераций, необходимых для достижения заданной точности увеличивается.

***Задача 2***

Решить задачу 1 с использованием для решения СЛАУ метод SOR.

Решение:

Параметр релаксации выбирается фиксированным и равным 1,9.

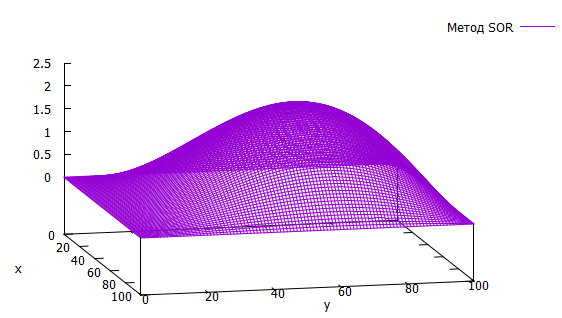


Рисунок 5. График численного решения

|  |  |  |
| --- | --- | --- |
| **N** | **Итерации** | **Погрешность** |
| 10 | 222 | 0,0178189 |
| 30 | 215 | 0,00197153 |
| 50 | 230 | 0,000710153 |
| 70 | 431 | 0,000362375 |
| 100 | 970 | 0,000177537 |

Таблица 2

По таблице 2 построим графики зависимостей от шага сетки погрешности решения методом релаксации и количества итераций.

Погрешность

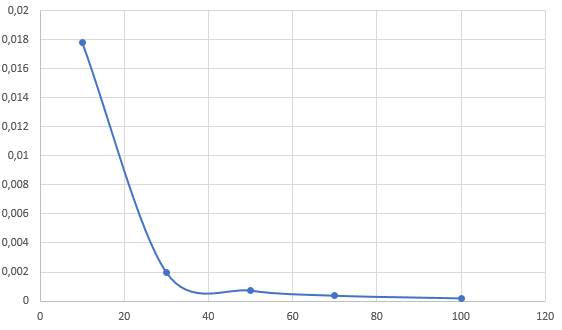
N

Рисунок 6.График зависимости погрешности от шага сетки

Кол-во итераций

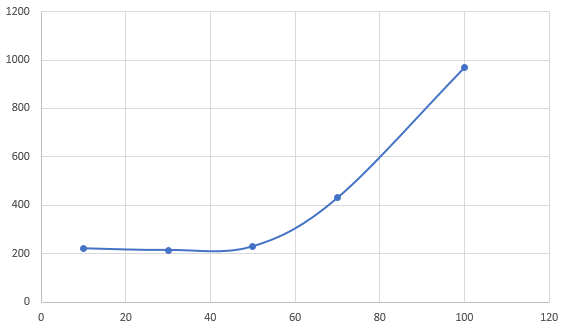
N

Рисунок 7. График зависимости числа итераций от шага сетки

По полученным графикам можно сказать, что с увеличением количества узлов сетки погрешность вычислений уменьшается, а количество итераций растет, но гораздо меньше, чем для метода простых итераций, поэтому решение задачи методом SOR намного эффективнее, чем методом простых итераций при условии удачно подобранного коэффициента релаксации.

***Задача 3***

Решить задачу 1 с использованием для решения СЛАУ метода сопряженных градиентов.

Решение:

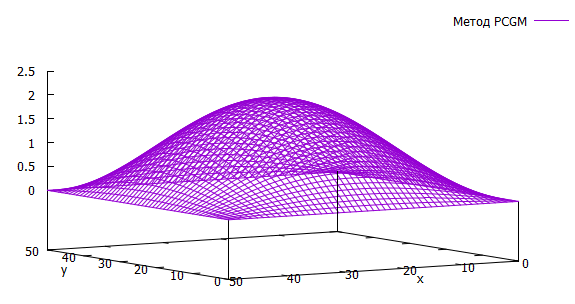


Рисунок 8.График численного решения

|  |  |  |
| --- | --- | --- |
| **N** | **Итерации** | **Погрешность** |
| 10 | 11 | 0,0223907 |
| 20 | 26 | 0,0055289 |
| 30 | 41 | 0,0024515 |
| 40 | 58 | 0,0013776 |
| 50 | 74 | 0,0008812 |

Таблица 3

По таблице 3 построим графики зависимостей от шага сетки погрешности решения методом релаксации и количества итераций.

Рисунок 9. График зависимости погрешности от шага сетки

Рисунок 10. График зависимости числа итераций от шага сетки

По полученным графикам можно сказать, что с увеличением количества узлов сетки погрешность вычислений уменьшается, но начиная с N=30 держится на одном уровне, а количество итераций растет.

**II. Решение задачи с переменными коэффициентами**

1. ***Задача 4.*** Написать вычислительную программу на языке программирования C++ решения задачи (1)-(2) методом переменных направлений.

|  |
| --- |
|  |
|  |

1. Исследовать зависимость погрешности получаемого решения от величины шага сетки, построить соответствующие графики.

**Вывод:**

В результате проделанной лабораторной работы был изучен теоретический материал необходимый для решения краевых задач для уравнений эллиптического типа с использованием различных методов на примере задачи Дирихле для линейного двумерного неоднородного уравнения.

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

**Приложение**

Листинг программы к задаче 1:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

const double epsilon = 1.0e-5;

double f(double x, double y)

{

return -M\_PI \* cos(M\_PI \* x / 2.) \* sin(M\_PI \* y) + 5. \* (x + 1.) \* M\_PI \* M\_PI \* sin(M\_PI \* x / 2.) \* sin(M\_PI \* y) / 4.;

}

double phi(double x, double y)

{

return 0.0;

}

double Solution(double x, double y)

{

return (x + 1.) \* sin(M\_PI \* x / 2.) \* sin(M\_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 Iterations(double\* y, double hx, double hy, int N)

{

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 <<"Iteration: "<< iter << endl;

}

int main()

{

int N = 70;

double lx = 2.0;

double ly = 1.0;

double hx = lx / N;

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

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

Iterations(y, hx, hy, N);

ofstream filewrite("Solutuon.txt");

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

{

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

filewrite << y[j \* (N + 1) + i] << '\t';

filewrite << endl;

}

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

return 0;

}

Листинг программы к задаче 2:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

double epsilon = 1.0e-10;

double f(double x, double y)

{

return -M\_PI \* cos(M\_PI \* x / 2.) \* sin(M\_PI \* y) + 5. \* (x + 1.) \* M\_PI \* M\_PI \* sin(M\_PI \* x / 2.) \* sin(M\_PI \* y) / 4.;

}

double phi(double x, double y)

{

return 0;

}

double Solution(double x, double y)

{

return (x + 1.)\* sin(M\_PI \* x / 2.)\* sin(M\_PI \* y);

}

double Error(double\* y, double hx, double hy, int N)

{

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(double\* y, double hx, double hy, int N, double w)

{

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 N = 10;

double lx = 2.0;

double ly = 1.0;

double hx = lx / N;

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

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

SOR(y, hx, hy, N, 1.9);

ofstream filewrite("SOR.txt");

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

{

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

{

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

}

filewrite << endl;

}

cout << Error(y, hx, hy, N) << endl;

return 0;

}

Листинг программы к задаче 3:

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

const double epsilon = 1.0e-10;

double f(double x, double y)

{

return -M\_PI \* cos(M\_PI \* x / 2.) \* sin(M\_PI \* y) + 5. \* (x + 1.) \* M\_PI \* M\_PI \* sin(M\_PI \* x / 2.) \* sin(M\_PI \* y) / 4.;

}

double phi(double x, double y)

{

return 0.0;

}

double Solution(double x, double y)

{

return (x + 1.) \* sin(M\_PI \* x / 2.) \* sin(M\_PI \* y);

}

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\* PCGM(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);

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 Error(double\* y, double hx, double hy, int N)

{

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 N = 100;

double lx = 2., 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 = PCGM(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("PCGM.txt");

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

{

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

filewrite << y[j \* (N + 1) + i] << '\t';

filewrite << endl;

}

cout << "Error:" <<Error(y, hx, hy, N) << endl;

return 0;

}

Листинг программы к задаче 4: