**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 τ для уравнения

$$\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)=-f(x,y)$$

Итерационный процесс в этом случае имеет вид

$$u_m^{n+1}=u_m^n+h\left(u_{m-1}^n-2u_m^n+u_{m+1}^n+\varphi_m^n\right);(3)$$

h-оптимальный выбор

$$h=\frac{h_x^2h_y^2}{2\left(h_x^2+h_y^2\right)}$$

Метод Якоби

Пусть $A=A^T\geq 0, a_{ij}\neq 0$. Представим A в виде $A=D-B$, где D – матрица с диагональными элементами A, а B – недиагональные элементы A. Тогда $Ax=b\Rightarrow Dx=Bx+b,\Rightarrow x=Hx+c, H=D^{-1}B, c=D^{-1}b$. Таким образом, метод Якоби – модификация метода простых итераций.

Координатная форма:

$$x_i^{k+1}=\frac{1}{a_{ii}}\left(-\sum_{j=1,j\neq i}^{n}a_{ij}x_j^k+b_i\right), i=1..n$$

Метод SOR

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

$$\begin{cases}b_1=a_{11}x_1+\ldots+a_{1n}x_n\\ b_2=a_{21}x_1+\ldots+a_{2n}x_n\\ \ldots\\ b_n=a_{n1}x_1+\ldots+a_{nn}x_n\end{cases}$$

Приводится к виду:

$$\begin{cases}b_{11}x_1+\ldots+b_{1n}x_n+c_1=0\\ \ldots\\ b_{n1}x_1+\ldots+b_{nn}x_n+c_n=0\end{cases}, где\, b_{ij}=\frac{-a_{ij}}{a_{ii}}, b_i=\frac{b_i}{a_{ii}}$$

Находится невязка

$$\begin{cases} R_1^{(0)}=c_1-x_1^{(0)}+\sum_{j=2}^{n} b_{1j} x_j^{(0)} \\ R_2^{(0)}=c_2-x_2^{(0)}+\sum_{j=1,\, j\neq 2}^{n} b_{2j} x_j^{(0)} \\ \dots \\ R_n^{(0)}=c_n-x_n^{(0)}+\sum_{j=1}^{n-1} b_{nj} x_j^{(0)} \end{cases}$$

Выбирается начальное приближение $X^0=0$. На каждом шаге необходимо обратить в ноль максимальную невязку: $R_s^{(k)}=\delta x_s^{(k)} \to R_s^{(k+1)}=0,\, R_i^{(k+1)}=R_i^{(k)}+b_{is}\,\delta x_s^{(k)}$

Ответ находится по формуле: $x_i \approx x_i^{(0)}+\sum_{j=2}^{n} \delta x_i^{(j)}$

$$\omega=\frac{2}{1+\sqrt{\mu}}$$

$$\mu=4\sin^2\frac{\pi h}{2}$$

y_{ij}^{n+1} находится из уравнения

$$\frac{y_{i-1j}^{n+1}+y_{ij-1}^{n+1}}{h^2}+\frac{y_{ij+1}^{n}+y_{i+1j}^{n}}{h^2}-\frac{4}{h^2}\left(\frac{1}{\omega}y_{ij}^{n+1}+\left(1-\frac{1}{\omega}\right)y_{ij}^{n}\right)=-f_{ij}$$

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.

Среди элементов матрицы А $a_{ij}, i, j=1\dots n$ выбирается наибольший по модулю a_{pq}, называемый главным элементом. Соответственно строка с этим элементом будет главной строкой. Предположим, что $a_{ij}=a_{11}$, 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 $a_{ij}=a_{11}$ и получают уравнение вида:

$$x_1+c_{12}x_2+\dots+c_{1n}x_n=d_1 \quad (1)$$

Где $c_{ij}=\frac{a_{1j}}{a_{11}}, d_1=\frac{b_1}{a_{11}}\ j=2..n$

На следующем шаге исключают, неизвестную величину x_1 из каждого уравнения исходной системы начиная со второго, путем вычитания (1), умноженного на коэффициент a_{ij}, при x_1 в соответствующем уравнении. Отбрасывают главную строку и первый столбец матрицы А и получают преобразованную систему уравнений

$$\begin{cases} c_{22}x_2+\dots+c_{2n}x_n=d_2 \\ \dots \\ c_{nn}x_n=d_n \end{cases} \quad (2)$$

Обратный ход нахождения коэффициентов через систему (1)

$$\begin{cases} x_n = \dfrac{d_n}{c_{nn}} \\ \cdots \\ x_1 = -c_{12}x_2 - \cdots - c_{1n}x_n - d_1 \end{cases} \qquad (2)$$

Практическая часть

Краевая задача для уравнения эллиптического типа

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

$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(b(x,y)\frac{\partial u}{\partial y}\right)+c(x,y)u+f(x,y)=0,(x,y)\in\Omega=(0,2)\times(0,1); \qquad (1)$$

$$u\vee ¿_{\Gamma}=\varphi(x,y),(x,y)\in\Gamma=\partial\Omega.¿ \qquad (2)$$

$$u(x,y)=(x+1)\sin\left(\frac{\pi x}{2}\right)\sin(\pi y)$$

I. Задача Дирихле для уравнения Пуассона с постоянными коэффициентами

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

$$a(x,y)=b(x,y)=1,c(x,y)=0. \qquad (3)$$

По заданному в индивидуальном задании точному решению задачи необходимо восстановить функции $f(x,y)$ и $\varphi(x,y)$.

$$\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)+f(x,y)=0,(x,y)\in\Omega=(0,2)\times(0,1); \qquad (1)$$

$$u\vee ¿_{\Gamma}=\varphi(x,y),(x,y)\in\Gamma=\partial\Omega.¿ \qquad (2)$$

$$u(x,y)=(x+1)\sin\left(\frac{\pi x}{2}\right)\sin(\pi y)$$

$$f(x,y)=-\pi cos\left(\frac{\pi x}{2}\right)\sin(\pi y)+\frac{5(x+1)\pi^2\sin\left(\frac{\pi x}{2}\right)\sin(\pi y)}{4},$$

$$\varphi(x,y)=0.$$

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 h_x и h_yпо направлениям *x* и *y*, удовлетворяющих соотношению
$$\frac{h_x}{h_y}=\frac{l_x}{l_y}.$$
Для решения получающейся СЛАУ использовать метод простых итераций. При этом матрица системы не должна храниться в памяти.
2) Исследовать зависимость погрешности решения от величины шагов сетки и построить соответствующие графики. Погрешность оценивать в равномерной норме.

3) Исследовать зависимости числа итераций от шага сетки.

2	1	$(x+1)\sin\left(\frac{\pi x}{l_x}\right)\sin\left(\frac{\pi y}{l_y}\right)$

<u>Решение:</u>

$$> plot3d\left((x+1)\cdot\sin\left(\frac{3.1415\cdot x}{2}\right)\cdot\sin(3.1415\cdot y), x=0..2, y=0..1\right)$$

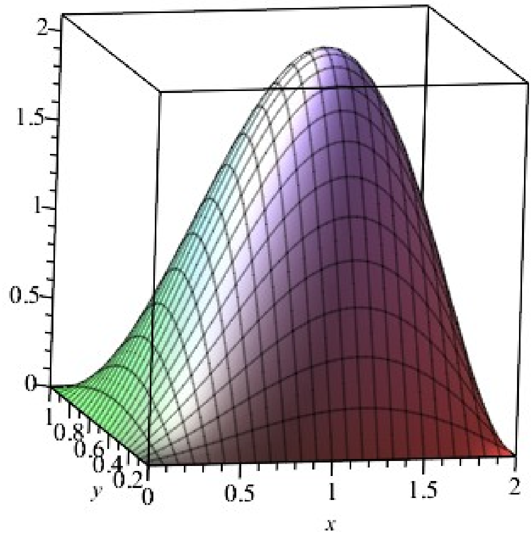


Рисунок 1. График точного решения

Условие остановки итерационного процесса следующее:

$$\|y_{k+1}-y_k\|\leq 10^{-10}.$$

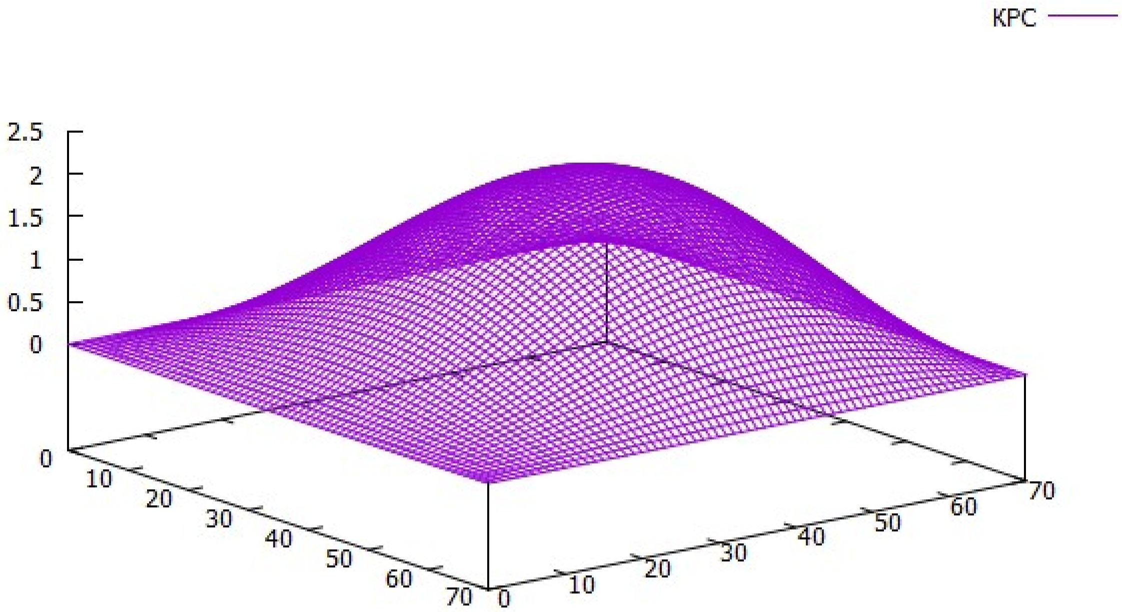


Рисунок 2. График численного решения

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

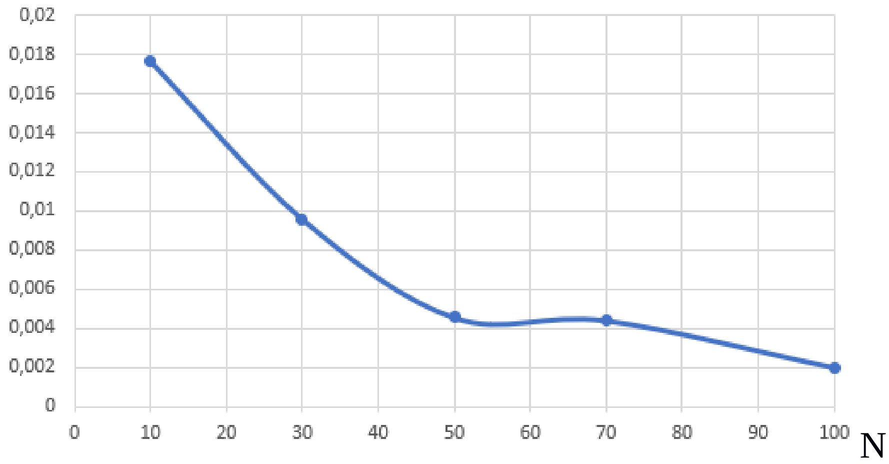


Рисунок 3. График зависимости погрешности от шага сетки

При увеличении числа узлов сетки погрешность уменьшается.

Кол-во итераций

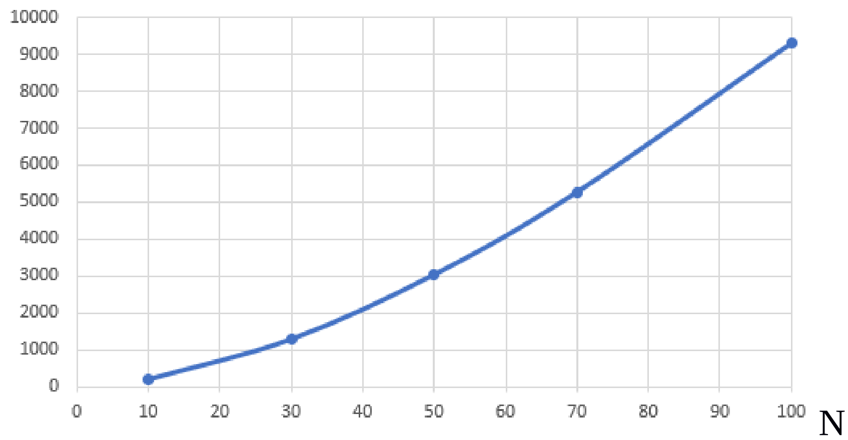


Рисунок 4. Зависимость числа итераций от шага сетки

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.

<u>Решение:</u>

Параметр релаксации выбирается фиксированным и равным 1,9.

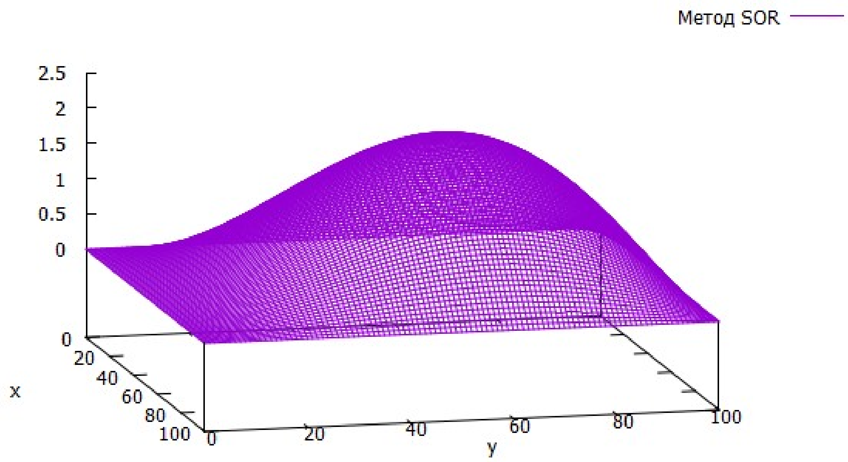


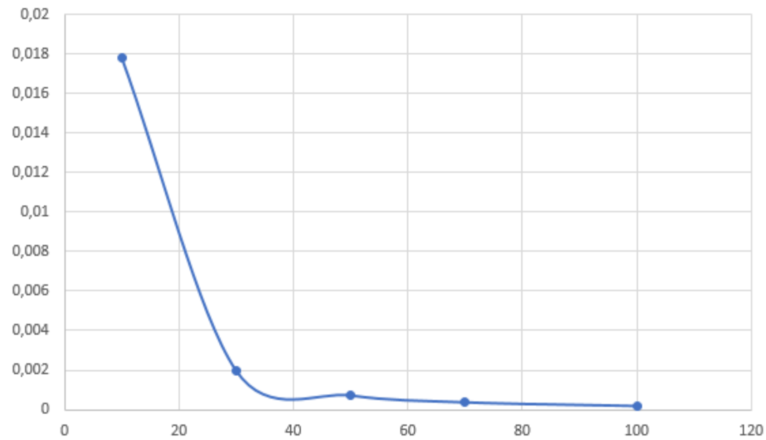
Рисунок 5. График численного решения

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

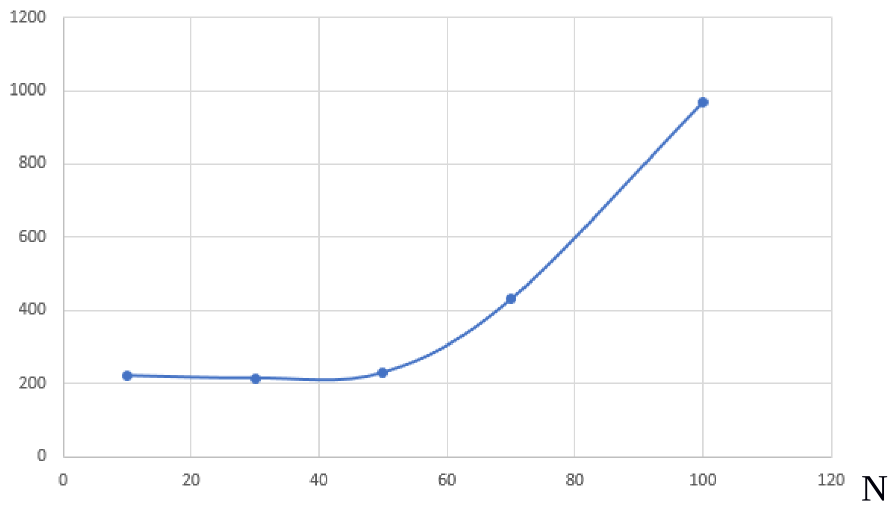


Рисунок 7. График зависимости числа итераций от шага сетки

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:

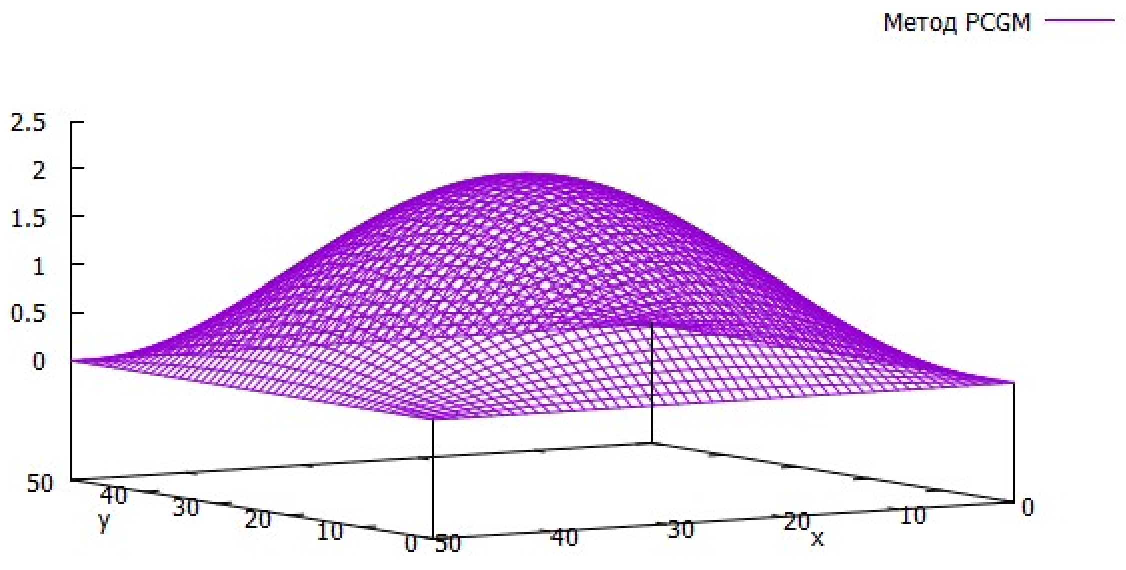


Рисунок 8.График численного решения

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.

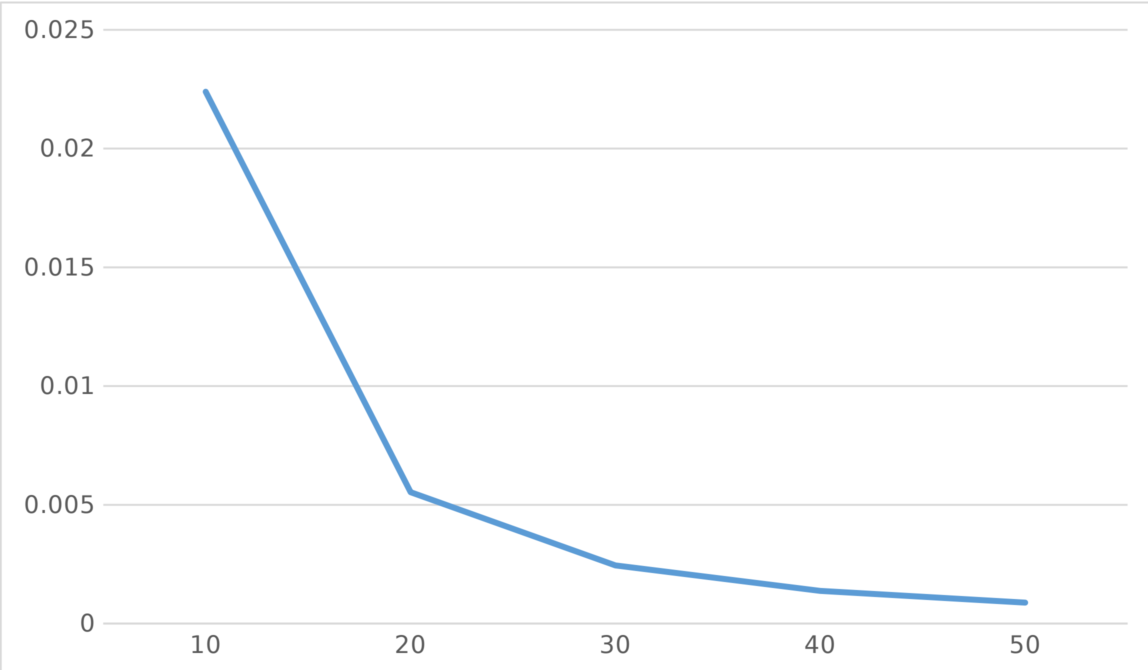


Рисунок 9. График зависимости погрешности от шага сетки

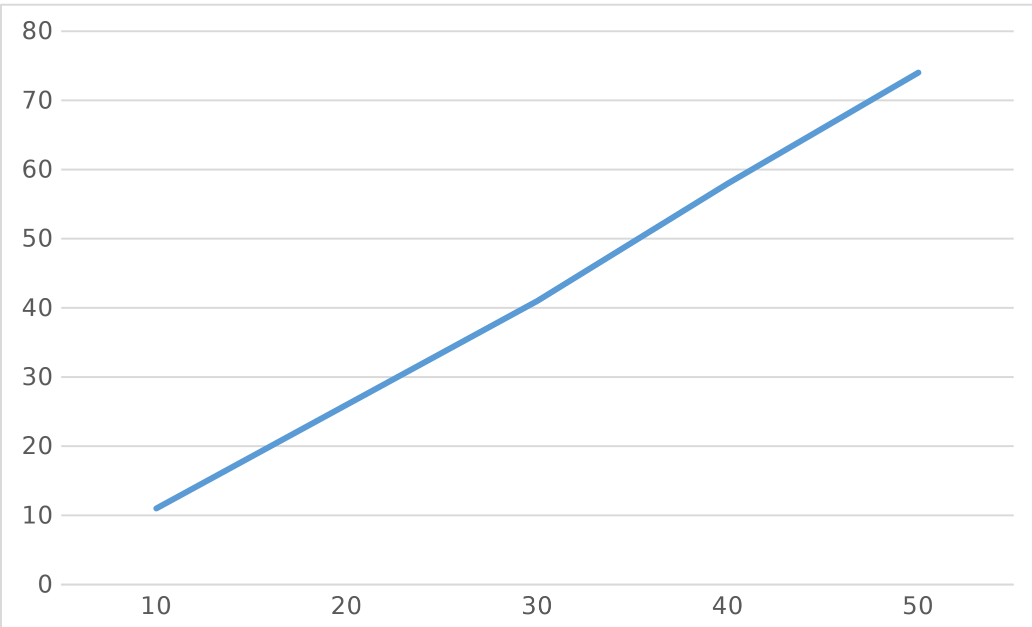


Рисунок 10. График зависимости числа итераций от шага сетки

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.

$$\frac{\partial}{\partial x}\left(y\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(x\frac{\partial u}{\partial y}\right)+u+f(x,y)=0,(x,y)\in\Omega=(0,2)\times(0,1);$$

$$u|_{\Gamma}=\varphi(x,y),(x,y)\in\Gamma=\partial\Omega.$$

$$u(x,y)=(x+1)\sin\left(\frac{\pi x}{2}\right)\sin(\pi y).$$

$$f(x,y)=-(x+1)\sin\left(\frac{\pi x}{2}\right)\sin(\pi y)-$$

$$-y\left(\pi\cos\left(\frac{\pi x}{2}\right)\sin(\pi y)-\frac{(x+1)\pi^2\sin\left(\frac{\pi x}{2}\right)\sin(\pi y)}{4}\right)+$$

$$+x\pi^2(x+1)\sin\left(\frac{\pi x}{2}\right)\sin(\pi y).$$

2) 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 C++, выполняющая необходимые построения и расчеты.

Приложение

Листинг программы к задаче 1:

```
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
```

```cpp
#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: