**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 МКН-316 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Чучкалов Д.А. Хромых К. М.. |  |  |  |
| Принял | Гайнетдинова А.А. |  |  |  |

**Уфа 2024**

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

Рассматривается начально-краевая задача для нелинейного одномерного уравнения теплопроводности с источником:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |

**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

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1 ошибки на различных шагах.

Изображение выглядит как зарисовка, шаблон, дизайн

Автоматически созданное описание

Рисунок 2. График аналитического решения.

Изображение выглядит как зарисовка, диаграмма, линия, шаблон

Автоматически созданное описание

Рисунок 3. График численного решения.

***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.

Изображение выглядит как зарисовка, диаграмма, линия, шаблон

Автоматически созданное описание

Рисунок 4. График численного решения

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Рисунок 5. Ошибки на различных шагах сетки.

Решается нелинейная задача (1)-(4) с дополнительными исходными данными и

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |

***Задача 3.***

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

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 6. Ошибки на различных шагах сетки.(LU разложение)

Изображение выглядит как зарисовка, диаграмма, линия, дизайн

Автоматически созданное описание

Рисунок 7. График решения нелинейного уравнения

***Задача 4.***

1. Написать вычислительную программу на языке программирования C++ решения задачи (1)-(2) с параметрами из таблиц 1 и 2 методом переменных направлений, либо использовать другой достаточно метод решения СЛАУ (точный метод или метод сопряженных градиентов).
2. Исследовать зависимость погрешности получаемого решения от величины шага сетки, построить соответствующие графики.

Изображение выглядит как зарисовка, рисунок, дизайн

Автоматически созданное описание

Рисунок 8. График численного решения

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9. Ошибки на различных шагах сетки

Вывод: в ходе лабораторной работы был получен навык численного решения линейных и нелинейных начально-краевых задач для уравнений параболического типа с использованием различных конечно-разностных схем на примере задачи для одномерного уравнения теплопроводности с источником.

**Листинг**

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

#include <iomanip>

#include <Eigen/Sparse>,<Eigen/Dense>

#define pi 3.1415926535

using namespace std;

using namespace Eigen;

double u\_analitic(double x, double y)

{

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

}

void createAnaliticFile(int N, int M)

{

ofstream ftc("analitic.txt");

double hx = 1. / (N - 1), hy = 1. / (M - 1);

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

ftc << u\_analitic(hx \* i, 0) << " ";//border y = 0;

}

ftc << endl;

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

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

if (i == 0) {

ftc << u\_analitic(0, hy \* j) << " ";//border x=0;

}

else if (i == N) {

ftc << u\_analitic(hx \* i,hy \* j) << " ";//border x = n;

}

else

ftc << u\_analitic(i \* hx, j \* hy) << " "; // matrix\_output

}

ftc << endl;

}

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

ftc << u\_analitic(hx \* i, 1) << " "; //border y = n;

}

}

double a(double x, double y)

{

return exp(x);

}

double b(double x, double y)

{

return exp(y);

}

double c(double x, double y)

{

return 1;

}

double phi1(double y) { //x=0

return 0;

}

double phi2(double y) { // x=lx

return 0;

}

double phi3(double x) { // y=0

return 0;

}

double phi4(double x) { // y=ly

return 0;

}

double func(double x, double y) {

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

}

double f\_4(double x, double y) {

return (-6.283185308) \* exp(x) \* sin(3.141592654 \* x) \* sin(3.141592654 \* y) \* cos(3.141592654 \* x) - 19.73920881 \* exp(x) \* cos(3.141592654 \* x) \* cos(3.141592654 \* x) \* sin(3.141592654 \* y) + 19.73920881 \* exp(x) \* sin(3.141592654 \* x) \* sin(3.141592654 \* x) \* sin(3.141592654 \* y) - 3.141592654 \* exp(y) \* sin(3.141592654 \* x) \* sin(3.141592654 \* x) \* cos(3.141592654 \* y) + 9.869604404 \* exp(y) \* sin(3.141592654 \* x) \* sin(3.141592654 \* x) \* sin(3.141592654 \* y) - 1. \* sin(3.141592654 \* x) \* sin(3.141592654 \* x) \* sin(3.141592654 \* y);

}

double U\_matrix\_returning(int i, int j, int N, int M) {

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

double hy = 1. / (M - 1);

double alpha = -2. \* (1. / hx / hx + 1. / hy / hy);

double betta = 1. / hx / hx;

double gamma = 1. / hy / hy;

if (i == j) {

return alpha;

}

else if (i - j == 1 && i % (N - 1) != 0) {

return betta;

}

else if (i - j == -1 && j % (N - 1) != 0) {

return betta;

}

else if (abs(i - j) == N - 1) {

return gamma;

}

else

return 0.;

}

void fill\_nonlinear\_matrix(int i, int j, int N, int M) {

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

double hy = 1. / (M - 1);

vector<double> left\_gamma;

vector<double> right\_gamma;

vector<double> left\_betta;

vector<double> right\_betta;

vector<double> alpha;

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

{

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

{

left\_gamma.push\_back(-b(i \* hx, j \* hy) / 2. / hy + b(i \* hx, j \* hy) / hy / hy);

right\_gamma.push\_back(b(i \* hx, j \* hy) / 2. / hy + b(i \* hx, j \* hy) / hy / hy);

alpha.push\_back(-2. \* a(i \* hx, j \* hy) / hx / hx + b(i \* hx, j \* hy) / hy / hy + c(i \* hx, j \* hy));

left\_betta.push\_back(-a(i \* hx, j \* hy) / 2 / hx + a(i \* hx, j \* hy) / hx / hx);

right\_betta.push\_back(a(i \* hx, j \* hy) / 2 / hx + a(i \* hx, j \* hy) / hx / hx);

}

}

}

vector<double> Matrix\_vector\_Multi(int N, int M, vector<double>x) {

vector<double> ax((N - 1) \* (M - 1));

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

double hy = 1. / (M - 1);

double betta = 1. / hx / hx;

double gamma = 1. / hy / hy;

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

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

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

if (i > 0) {

ax[j \* (N - 1) + i] += betta \* x[j \* (N - 1) + i - 1]; // betta left

}

if (i < N - 2) {

ax[j \* (N - 1) + i] += betta \* x[j \* (N - 1) + i + 1]; //betta right

}

if (j < M - 2) {

ax[j \* (N - 1) + i] += gamma \* x[(j + 1) \* (N - 1) + i]; // gamma right

}

if (j > 0) {

ax[j \* (N - 1) + i] += gamma \* x[(j - 1) \* (N - 1) + i]; // gamma left

}

//\*Idea: we are skippin all invalid memory operetions via including if-statements and summing all by parts:

}

}

return ax;

}

void filling\_F(vector<double>& f, int N, int M, int paramtoZadanie = 0) {

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

double hy = 1. / (M - 1);

if (paramtoZadanie != 0) {

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

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

f[j \* (N - 1) + i] = f\_4((i + 1) \* hx, (j + 1) \* hy);

if (j == 0) {

f[j \* (N - 1) + i] -= phi3((i + 1) \* hx) / hy / hy;

}

if (j == (M - 1)) {

f[j \* (N - 1) + i] -= phi4((i + 1) \* hx) / hy / hy;

}

if (i == (N - 1)) {

f[j \* (N - 1) + i] -= phi2((j + 1) \* hy) / hx / hx;

}

if (i == 0) {

f[j \* (N - 1) + i] -= phi1((j + 1) \* hy) / hx / hx;

}

}

}

}

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

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

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

if (j == 0) {

f[j \* (N - 1) + i] -= phi3((i + 1) \* hx) / hy / hy;

}

if (j == (M - 1)) {

f[j \* (N - 1) + i] -= phi4((i + 1) \* hx) / hy / hy;

}

if (i == (N - 1)) {

f[j \* (N - 1) + i] -= phi2((j + 1) \* hy) / hx / hx;

}

if (i == 0) {

f[j \* (N - 1) + i] -= phi1((j + 1) \* hy) / hx / hx;

}

}

}

}

void filling\_F(VectorXd& f, int N, int M, int paramtoZadanie = 0) {

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

double hy = 1. / (M - 1);

if (paramtoZadanie != 0) {

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

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

f[j \* (N - 1) + i] = -f\_4((i + 1) \* hx, (j + 1) \* hy);

if (j == 0) {

f[j \* (N - 1) + i] -= phi3((i + 1) \* hx) / hy / hy;

}

if (j == (M - 1)) {

f[j \* (N - 1) + i] -= phi4((i + 1) \* hx) / hy / hy;

}

if (i == (N - 1)) {

f[j \* (N - 1) + i] -= phi2((j + 1) \* hy) / hx / hx;

}

if (i == 0) {

f[j \* (N - 1) + i] -= phi1((j + 1) \* hy) / hx / hx;

}

}

}

}

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

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

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

if (j == 0) {

f[j \* (N - 1) + i] -= phi3((i + 1) \* hx) / hy / hy;

}

if (j == (M - 1)) {

f[j \* (N - 1) + i] -= phi4((i + 1) \* hx) / hy / hy;

}

if (i == (N - 1)) {

f[j \* (N - 1) + i] -= phi2((j + 1) \* hy) / hx / hx;

}

if (i == 0) {

f[j \* (N - 1) + i] -= phi1((j + 1) \* hy) / hx / hx;

}

}

}

}

void LU\_decompostion(int N, int M, vector<double> &L, vector<double> &U)

{

int n = (N - 1) \* (M - 1);

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

{

L[i \* n] = U\_matrix\_returning(i, 0, N, M);

U[i] = U\_matrix\_returning(i, 0, N, M) / L[0];

}

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

{

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

{

U[i \* n + j] = U\_matrix\_returning(i, j,N,M);

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

{

U[i \* n + j] -= L[i \* n + k] \* U[k \* n + j];

}

L[j \* n + i] = U\_matrix\_returning(i, j, N, M);

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

{

L[j \* n + i] -= L[j \* n + k] \* U[k \* n + i];

}

L[j \* n + i] /= U[i \* n + i];

}

}

}

void LU\_decompostion1(int N, int M, vector<double>& L, vector<double>& U)

{

int n = (N - 1) \* (M - 1);

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

{

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

{

U[i \* n + j] = U\_matrix\_returning(i, j, N, M);

}

}

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

{

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

{

L[j \* n + i] = U[j \* n + i] / U[i \* n + i];

}

}

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

{

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

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

L[j \* n + i] = U[j \* n + i] / U[i \* n + i];

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

for (int j = k - 1; j < n; j++)

U[i \* n + j] = U[i \* n + j] - L[i \* n + k - 1] \* U[(k - 1) \* n + j];

}

}

vector<double> backward\_up(int N, int M, vector<double> &U, vector<double> &b)

{

int n = (N - 1) \* (M - 1);

vector<double> X(n \* n);

for (int i = n - 1; i >= 0; i--)

{

X[i] = b[i];

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

{

X[i] -= U[i \* n + j] \* X[j];

}

X[i] /= U[i \* n + i]; // /= должно быть

}

return X;

}

vector<double> backward\_low(int N, int M, vector<double> &L, vector<double> &b)

{

int n = (N - 1) \* (M - 1);

vector<double> Y(n \* n);

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

{

Y[i] = b[i];

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

{

Y[i] -= L[i \* n + j] \* Y[j];

}

Y[i] /= L[i + i \* n];

}

return Y;

}

vector<double> solve\_LU(int N, int M, vector<double> &b)

{

int n = (N - 1) \* (M - 1);

vector<double> L(n \* n, 0), U(n \* n, 0);

LU\_decompostion(N, M, L, U);

auto Y = backward\_low(N, M, L, b);

auto X = backward\_up(N, M, U, Y);

return X;

}

double vec\_vec\_dot(vector<double>& rhs, vector<double>& lhs) {

double result = 0;

for (int i = 0; i < rhs.size(); i++) {

result += rhs[i] \* lhs[i];

}

return result;

}

vector<double> vectorSubtraction(vector<double> &A, vector<double> &B)

{

vector<double> result(A.size(), 0);

for (int i = 0; i < A.size(); i++)

{

result[i] = A[i] - B[i];

}

return result;

}

void z1\_Poisson\_problem\_eq(int N, int M, int paramToCout = 0) {

double hx = 1. / (N - 1), hy = 1. / (M - 1);

vector<double> f((N - 1) \* (M - 1));

vector<double> x\_prev((N - 1) \* (M - 1), 0);

double alpha = -2 \* (1. / hx / hx + 1. / hy / hy);

vector<double> x\_next((N - 1) \* (M - 1));

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

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

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

if (j == 0) {

f[j \* (N - 1) + i] -= phi3((i + 1) \* hx) / hy / hy;

}

if (j == (M - 1)) {

f[j \* (N - 1) + i] -= phi4((i + 1) \* hx) / hy / hy;

}

if (i == (N - 1)) {

f[j \* (N - 1) + i] -= phi2((j + 1) \* hy) / hx / hx;

}

if (i == 0) {

f[j \* (N - 1) + i] -= phi1((j + 1) \* hy) / hx / hx;

}

}

}

// x\_k+1 = b - c\*x\_k; c = {aij,ii=0;}

double diff = -1;

x\_next = f;

vector<double> buf\_Mult\_MV;

while (abs(diff) > 1e-15) {

x\_prev = x\_next;

diff = 0;

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

buf\_Mult\_MV = Matrix\_vector\_Multi(N, M, x\_prev);

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

x\_next[j \* (N - 1) + i] = (f[j \* (N - 1) + i] - buf\_Mult\_MV[j \* (N - 1) + i]) / U\_matrix\_returning(i, i, N, M);

diff = max(abs(x\_prev[j \* (N - 1) + i] - x\_next[j \* (N - 1) + i]), diff);

}

}

//cout << diff << endl;

}

double diff\_analitic = -1;

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

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

diff\_analitic = max(abs(u\_analitic((i + 1) \* hx, (j + 1) \* hy) - x\_next[j \* (N - 1) + i]), diff\_analitic); //err estimation

}

}

cout << setprecision(3) << " hx: " << hx << " hy: " << hy << " Error: " << diff\_analitic << endl;

if (paramToCout == 1) {

ofstream ftc("task1.txt");

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

ftc << phi3(hx \* i) << " ";//border y = 0;

}

ftc << endl;

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

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

if (i == 0) {

ftc << phi1(hy \* j) << " ";//border x=0;

}

else if (i == N) {

ftc << phi2(hy \* j) << " ";//border x = n;

}

else

ftc << x\_next[j \* (N - 1) + i - 1] << " "; // matrix\_output

}

ftc << endl;

}

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

ftc << phi4(hx \* i) << " "; //border y = n;

}

}

}

void z2\_Poisson\_problem\_eq(int N, int M, int paramToCout = 0) {

double hx = 1. / (N - 1), hy = 1. / (M - 1);

vector<double> f((N - 1) \* (M - 1));

vector<double> x\_prev((N - 1) \* (M - 1), 0);

double alpha = -2 \* (1. / hx / hx + 1. / hy / hy);

double betta = 1. / hx / hx;

double gamma = 1. / hy / hy;

vector<double> x\_next((N - 1) \* (M - 1));

filling\_F(f, N, M);

// x\_k+1 = (1-omega)x\_k - omega/aii (b - s1 -s2); c = {aij,ii=0;}

double diff = -1;

x\_next = f;

double omega = 1.5; //SOR parameter

vector<double> buf\_Mult\_MV;

while (abs(diff) > 1e-15) {

x\_prev = x\_next;

diff = 0;

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

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

x\_next[j \* (N - 1) + i] = f[j \* (N - 1) + i];

if (i > 0) {

x\_next[j \* (N - 1) + i] -= betta \* x\_next[j \* (N - 1) + i - 1]; // betta left

}

if (i < N - 2) {

x\_next[j \* (N - 1) + i] -= betta \* x\_prev[j \* (N - 1) + i + 1]; //betta right

}

if (j < M - 2) {

x\_next[j \* (N - 1) + i] -= gamma \* x\_prev[(j + 1) \* (N - 1) + i]; // gamma right

}

if (j > 0) {

x\_next[j \* (N - 1) + i] -= gamma \* x\_next[(j - 1) \* (N - 1) + i]; // gamma left

}

x\_next[j \* (N - 1) + i] \*= omega / U\_matrix\_returning(i, i, N, M);

x\_next[j \* (N - 1) + i] += (1 - omega) \* x\_prev[j \* (N - 1) + i];

diff = max(abs(x\_prev[j \* (N - 1) + i] - x\_next[j \* (N - 1) + i]), diff);

}

}

//cout << diff << endl;

}

double diff\_analitic = -1;

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

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

diff\_analitic = max(abs(u\_analitic((i + 1) \* hx, (j + 1) \* hy) - x\_next[j \* (N - 1) + i]), diff\_analitic); //error estimation

}

}

cout << setprecision(3) << " hx: " << hx << " hy: " << hy << " Error: " << diff\_analitic << endl;

if (paramToCout == 1) {

ofstream ftc("task2.txt");

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

ftc << phi3(hx \* i) << " ";//border y = 0;

}

ftc << endl;

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

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

if (i == 0) {

ftc << phi1(hy \* j) << " ";//border x=0;

}

else if (i == N) {

ftc << phi2(hy \* j) << " ";//border x = n;

}

else

ftc << x\_next[j \* (N - 1) + i - 1] << " "; // matrix\_output

}

ftc << endl;

}

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

ftc << phi4(hx \* i) << " "; //border y = n;

}

}

}

void z3(int N, int M, int ptc)

{

vector<double> f((N - 1) \* (M - 1));

double hx = 1. / (N - 1), hy = 1. / (M - 1);

filling\_F(f, N, M);

auto solve = solve\_LU(N, M, f);

double diff\_analitic = -1;

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

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

diff\_analitic = max(abs(u\_analitic((i + 1) \* hx, (j + 1) \* hy) - solve[j \* (N - 1) + i]), diff\_analitic);

}

}

cout << setprecision(3) << " hx: " << hx << " hy: " << hy << " Error: " << diff\_analitic << endl;

if (ptc == 1)

{

ofstream ftc("task3.txt");

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

ftc << phi3(hx \* i) << " ";//border y = 0;

}

ftc << endl;

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

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

if (i == 0) {

ftc << phi1(hy \* j) << " ";//border x=0;

}

else if (i == N) {

ftc << phi2(hy \* j) << " ";//border x = n;

}

else

ftc << solve[j \* (N - 1) + i - 1] << " "; // matrix\_output

}

ftc << endl;

}

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

ftc << phi4(hx \* i) << " "; //border y = n;

}

}

}

void z4\_Poisson\_problem\_eq\_notwork(int N, int M, int paramToCout = 0) {

//cgm

double hx = 1. / (N - 1), hy = 1. / (M - 1);

int iter = 0;

double alpha = 0;

double betta = 0;

double err = 0; double x\_buf = 0;

vector<double> r\_prev((N - 1) \* (M - 1));

vector<double> f((N - 1) \* (M - 1));

filling\_F(f, N, M, 1);

filling\_F(r\_prev, N, M, 1);

vector<double> x((N - 1) \* (M - 1), 0.);

vector<double> r\_next((N - 1) \* (M - 1));

vector<double>Az((N - 1) \* (M - 1));

vector<double>z\_prev = r\_prev;

double x\_coord = 0;

double y\_coord = 0;

do {

err = 0;

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

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

x\_coord = hx \* (i + 1);

y\_coord = hy \* (j + 1);

Az[j \* (N - 1) + i] = (1 - 2 \* (exp(x\_coord) / hx / hx + exp(y\_coord) / hy / hy)) \* z\_prev[j \* (N - 1) + i];

if (i > 0) {

Az[j \* (N - 1) + i] += exp(x\_coord) \* (1. / hx / hx - 1. / 2. / hx) \* z\_prev[j \* (N - 1) + i - 1]; // betta left

}

if (i < N - 2) {

Az[j \* (N - 1) + i] += exp(x\_coord) \* (1. / hx / hx + 1. / 2. / hx) \* z\_prev[j \* (N - 1) + i + 1]; //betta right

}

if (j < M - 2) {

Az[j \* (N - 1) + i] += exp(y\_coord) \* (1. / hy / hy + 1. / 2. / hy) \* z\_prev[(j + 1) \* (N - 1) + i]; // gamma right

}

if (j > 0) {

Az[j \* (N - 1) + i] += exp(y\_coord) \* (1. / hy / hy - 1. / 2. / hy) \* z\_prev[(j - 1) \* (N - 1) + i]; // gamma left

}

}

}

alpha = vec\_vec\_dot(r\_prev, r\_prev) / vec\_vec\_dot(Az, z\_prev);

for (int i = 0; i < x.size(); i++) {

x\_buf = x[i];

x[i] += alpha \* z\_prev[i];

r\_next[i] = r\_prev[i] - alpha \* Az[i];

err = max(err, abs(x[i] - x\_buf));

}

betta = vec\_vec\_dot(r\_next, r\_next) / vec\_vec\_dot(r\_prev, r\_prev);

for (int i = 0; i < x.size(); i++) {

z\_prev[i] = r\_next[i] + betta \* z\_prev[i];

}

iter++;

//cout << sqrt(vec\_vec\_dot(r\_next, r\_next)) / sqrt(vec\_vec\_dot(f, f)) << "\tIter:" << iter <<endl;

r\_prev = r\_next;

} while (err > 1e-15);

double diff\_analitic = -1;

double max\_num = 0;

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

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

diff\_analitic = max(abs(u\_analitic((i + 1) \* hx, (j + 1) \* hy) - x[j \* (N - 1) + i]), diff\_analitic); //error estimation

max\_num = max(abs(x[j \* (N - 1) + i]), max\_num);

}

}

//cout << "MCGM" << endl;

cout << setprecision(3) << " hx: " << hx << " hy: " << hy << " Error: " << diff\_analitic << endl;

}

void z4\_Eigen(int N, int M, int paramToCout = 0) {

double hx = 1. / (N - 1), hy = 1. / (M - 1);

VectorXd f((N - 1) \* (M - 1));

filling\_F(f, N, M, 1);

double value = 0;

double x\_coord, y\_coord;

vector<Triplet<double>> triplets;

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

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

x\_coord = hx \* (i + 1);

y\_coord = hy \* (j + 1);

triplets.push\_back(Triplet<double>(j \* (N - 1) + i, j \* (N - 1) + i, (1 - 2 \* (exp(x\_coord) / hx / hx + exp(y\_coord) / hy / hy)))); // alpha

}

}

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

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

x\_coord = hx \* (i + 1);

y\_coord = hy \* (j + 1);

triplets.push\_back(Triplet<double>(j \* (N - 1) + i, j \* (N - 1) + i - 1, exp(x\_coord) \* (1. / hx / hx - 1. / 2. / hx))); // betta left

}

}

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

for (int i = 0; i < (N - 2); i++) {

x\_coord = hx \* (i + 1);

y\_coord = hy \* (j + 1);

triplets.push\_back(Triplet<double>(j \* (N - 1) + i, j \* (N - 1) + i + 1, exp(x\_coord) \* (1. / hx / hx + 1. / 2. / hx))); // betta right

}

}

for (int j = 0; j < (M - 2); j++) {

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

x\_coord = hx \* (i + 1);

y\_coord = hy \* (j + 1);

triplets.push\_back(Triplet<double>(j \* (N - 1) + i, (j + 1) \* (N - 1) + i, exp(y\_coord) \* (1. / hy / hy + 1. / 2. / hy))); // gamma right

}

}

for (int j = 1; j < (M - 1); j++) {

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

x\_coord = hx \* (i + 1);

y\_coord = hy \* (j + 1);

triplets.push\_back(Triplet<double>(j \* (N - 1) + i, (j - 1) \* (N - 1) + i, exp(y\_coord) \* (1. / hy / hy - 1. / 2. / hy))); // gamma left

}

}

SparseMatrix<double> A((N - 1) \* (M - 1), (N - 1) \* (M - 1));

A.setFromTriplets(triplets.begin(), triplets.end());

ConjugateGradient<SparseMatrix<double>, Lower | Upper> cg;

cg.compute(A);

auto x = cg.solve(f);

double diff\_analitic = -1;

double max\_num = 0;

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

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

diff\_analitic = max(abs(u\_analitic((i + 1) \* hx, (j + 1) \* hy) - x[j \* (N - 1) + i]), diff\_analitic); //error estimation

max\_num = max(abs(x[j \* (N - 1) + i]), max\_num);

}

}

//cout << "CG" << endl;

cout << setprecision(3) << " hx: " << hx << " hy: " << hy << " Error: " << diff\_analitic << endl;

}

int main() {

int Number\_of\_ex;

cin >> Number\_of\_ex;

vector <int> N = { 4,10,20,30,40,50,60 };

switch (Number\_of\_ex)

{

case 1:

for (auto& el : N)

{

z1\_Poisson\_problem\_eq(el, el);

}

//z1\_Poisson\_problem\_eq(30, 30,1);

break;

case 2:

for (auto& el : N)

{

z2\_Poisson\_problem\_eq(el, el);

}

//z2\_Poisson\_problem\_eq(30, 30, 1);

break;

case 3:

for (auto& el : N)

{

z3(el, el, 1);

}

case 4:

for (auto& el : N)

{

//z4\_Eigen(el, el);

z4\_Poisson\_problem\_eq\_notwork(el, el);

cout << "=====";

//Jacobi\_solver(el, el, 1);

}

//Jacobi\_solver(30, 30, 1);

case 5:

createAnaliticFile(N[N.size() - 1], N[N.size() - 1]);

break;

}

}