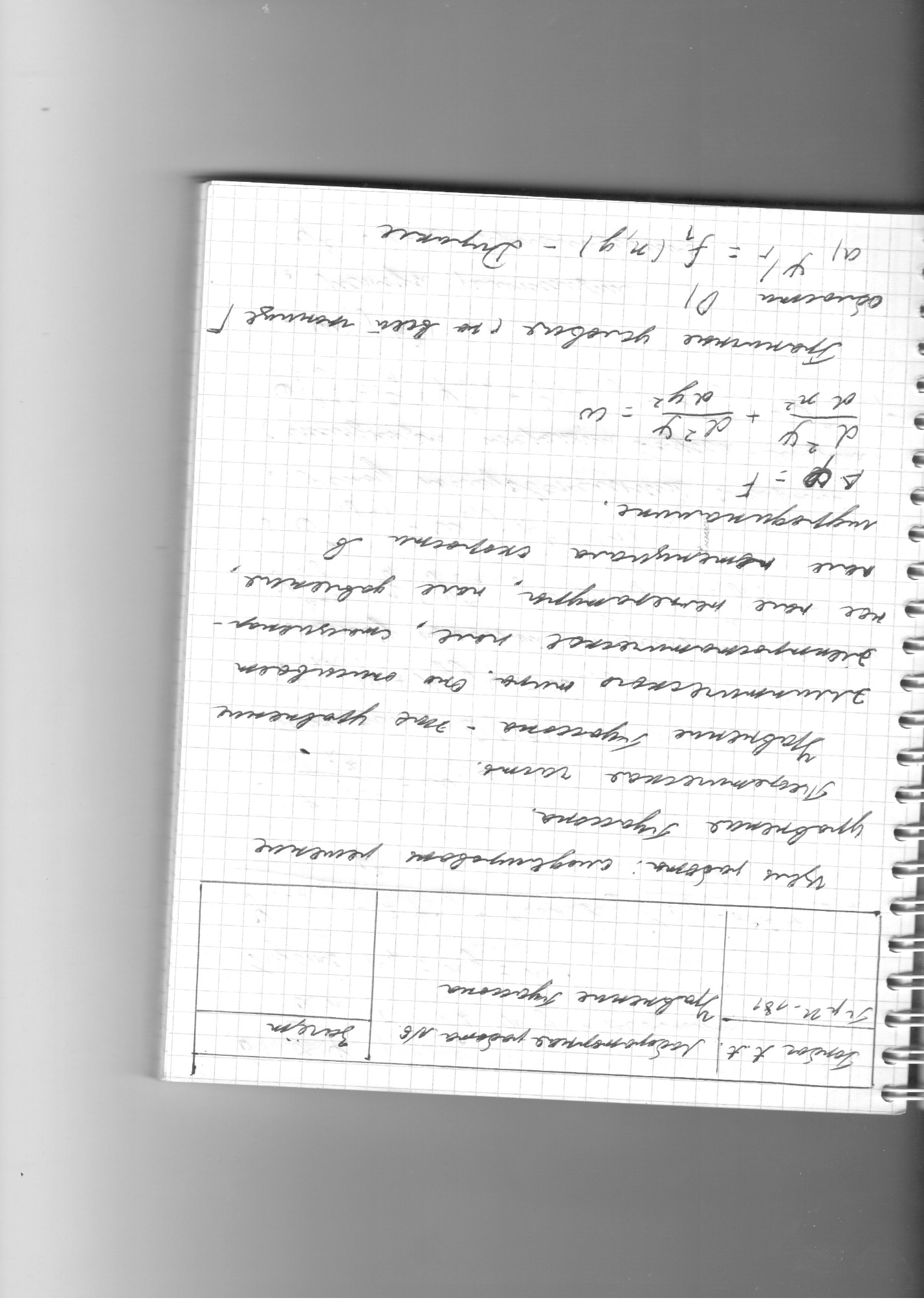
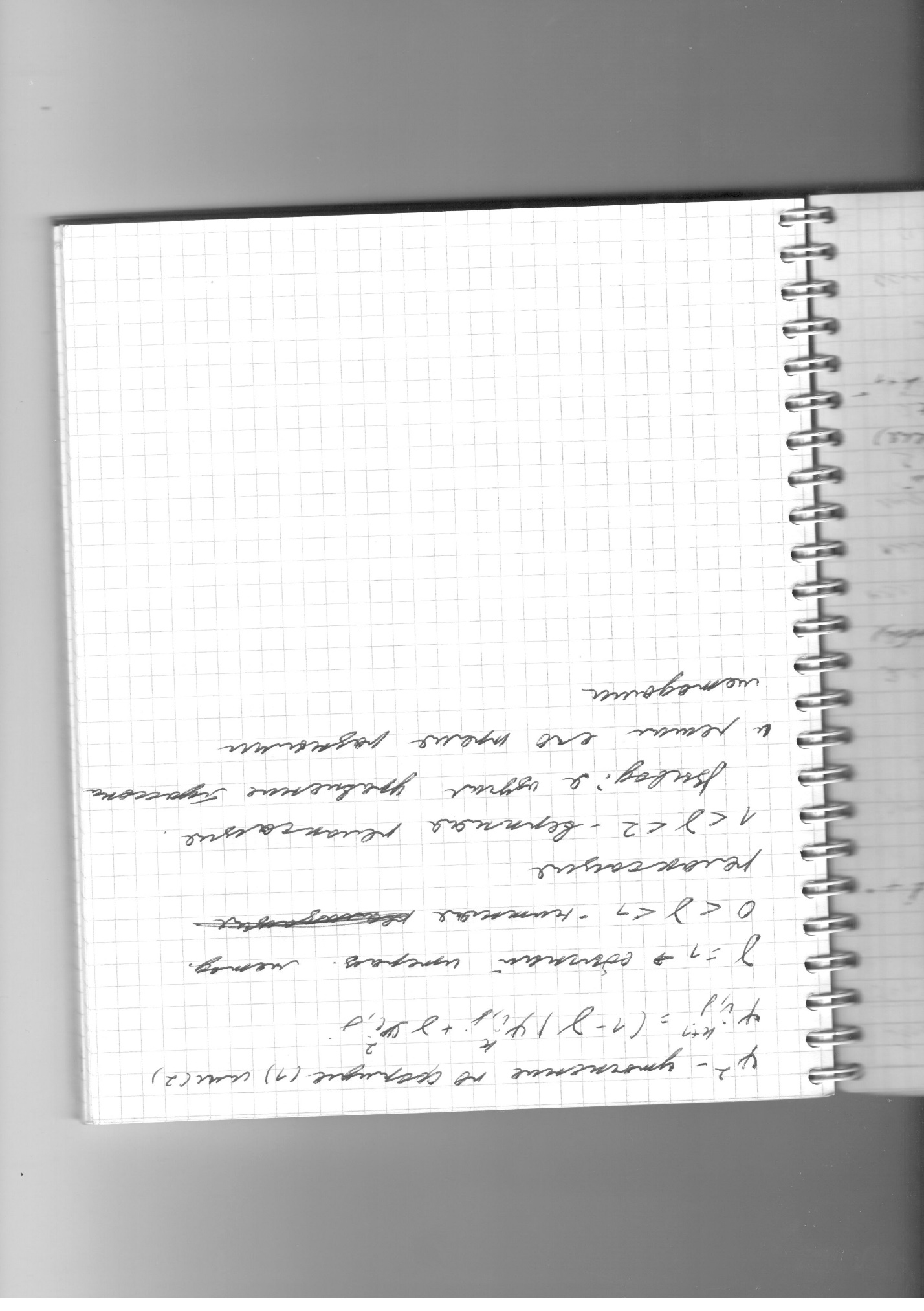
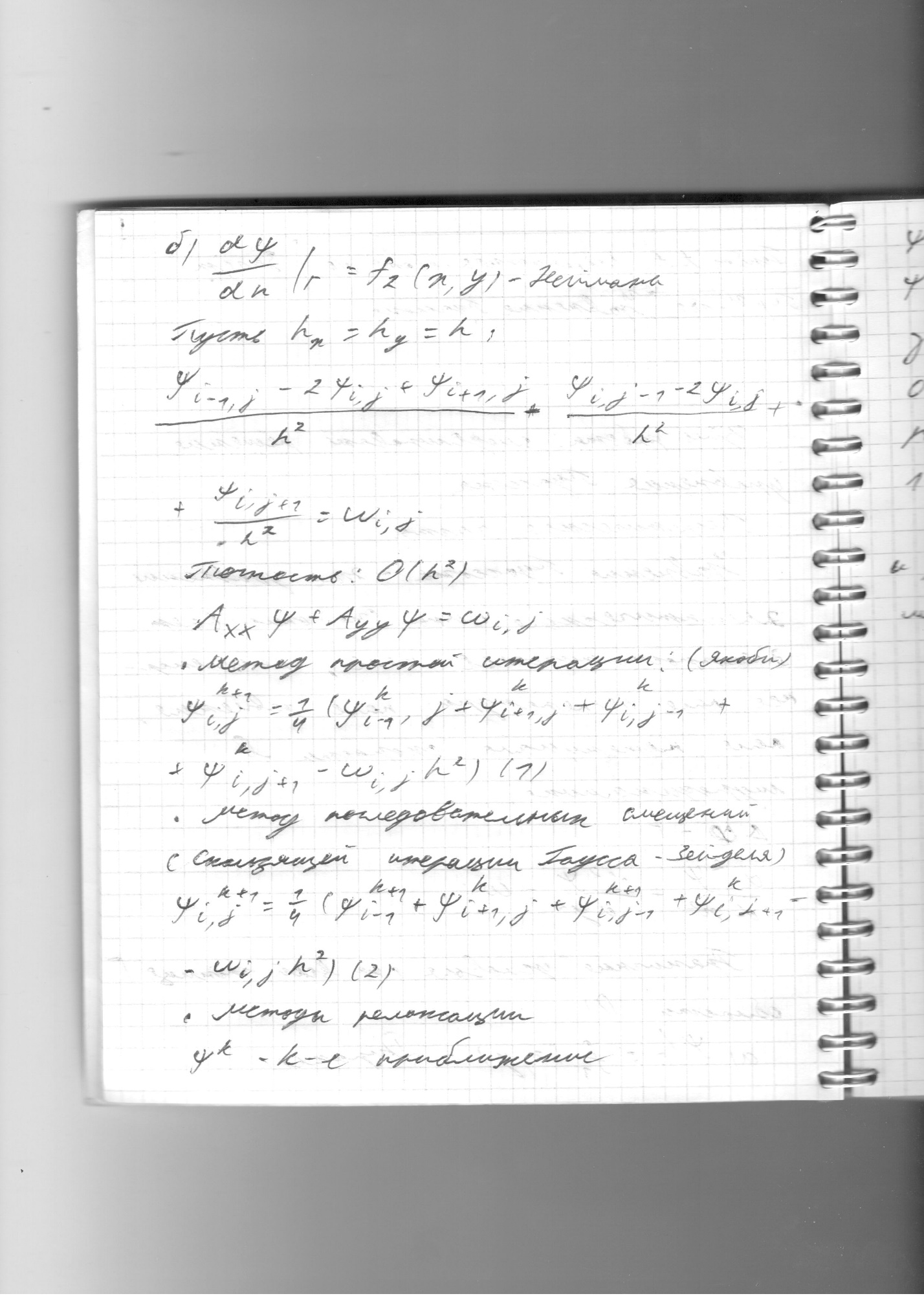
**Лабораторная работа №6.**

**Конспект.** 



**Практическая часть.**

Шаг dx = 0,5.

Граничные условия первого рода:

и

Рассчитываемая функция: .

Код программы:

#include <iostream>

#include <clocale>

#include <fstream>

#include <cmath>

#include <iomanip>

#include <chrono>

using namespace std;

double x\_min = 0;

double x\_max = 4;

double y\_min = 0;

double y\_max = 4;

double dx = 0.05;

double x, y;

double delta\_max;

double fi\_max;

double delta;

double fi\_tmp;

bool first = true;

double Omega(double x, double y)

{

return 2 \* x + y;

}

//Метод простых итераций

void Simple\_iter(double\*\* fi, double N, double M)

{

//Таймер (начало)

auto start = chrono::high\_resolution\_clock::now();

double\*\* fi\_next = new double\* [N + 1.0];

for (int i = 0; i < N + 1.0; i++)

{

fi\_next[i] = new double[M + 1.0];

}

//Занулление массива

for (int i = 0; i < N + 1.0; i++)

{

for (int j = 0; j < M + 1.0; j++)

{

fi\_next[i][j] = 0;

}

}

while (first == true || abs(delta\_max / fi\_max) > 0.00001)

{

first = false;

delta\_max = 0;

fi\_max = 0;

x = dx;

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

{

y = dx;

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

{

//Основная формула

fi\_next[i][j] = (fi[i - 1][j] + fi[i + 1][j] + fi[i][j - 1] + fi[i][j + 1] - Omega(x, y) \* pow(dx, 2)) / 4.0;

delta = fi\_next[i][j] - fi[i][j];

if (delta > delta\_max)

{

delta\_max = delta;

}

if (fi[i][j] > fi\_max)

{

fi\_max = fi[i][j];

}

y = y + dx;

}

x = x + dx;

}

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

{

//Меняем массивы местами

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

{

fi[i][j] = fi\_next[i][j];

}

}

}

//Таймер (конец)

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> duration = end - start;

cout << "\nВремя вычислений: " << std::setprecision(8) << std::fixed << duration.count() << endl;

//Загрузка в файл

ofstream file("simple.txt");

x = 0;

file << "x,y,fi" << endl;

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

{

y = 0;

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

{

file << x << ',' << y << ',' << fi[i][j] << endl;

y = y + dx;

}

x = x + dx;

}

cout << "\nВычисления успешно загружены в файл: simple.txt\n" << endl;

//Очистка памяти

for (int i = 0; i < N + 1.0; i++)

{

delete[] fi\_next[i];

}

}

//Метод последовательных смещений

void Successive\_disp(double\*\* fi, double N, double M)

{

auto start = chrono::high\_resolution\_clock::now();

while (first == true || abs(delta\_max / fi\_max) > 0.00001)

{

first = false;

delta\_max = 0;

fi\_max = 0;

x = dx;

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

{

y = dx;

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

{

//Основная формула

fi\_tmp = (fi[i - 1][j] + fi[i + 1][j] + fi[i][j - 1] + fi[i][j + 1] - Omega(x, y) \* pow(dx, 2)) / 4.0;

delta = fi\_tmp - fi[i][j];

if (delta > delta\_max)

{

delta\_max = delta;

}

if (fi[i][j] > fi\_max)

{

fi\_max = fi[i][j];

}

fi[i][j] = fi\_tmp;

y = y + dx;

}

x = x + dx;

}

}

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> duration = end - start;

cout << "\nВремя вычислений: " << std::setprecision(8) << std::fixed << duration.count() << endl;

ofstream file("successive\_disp.txt");

x = 0;

file << "x,y,fi" << endl;

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

{

y = 0;

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

{

file << x << ',' << y << ',' << fi[i][j] << endl;

y = y + dx;

}

x = x + dx;

}

cout << "\nВычисления успешно загружены в файл: successive\_disp.txt\n" << endl;

}

//Метод релаксации

void Relaxation(double gamma, double\*\* fi, double N, double M)

{

auto start = chrono::high\_resolution\_clock::now();

while (first == true || abs(delta\_max / fi\_max) > 0.00001)

{

first = false;

delta\_max = 0;

fi\_max = 0;

x = dx;

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

{

y = dx;

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

{

//Приближение по формуле последовательных смещений

fi\_tmp = (fi[i - 1][j] + fi[i + 1][j] + fi[i][j - 1] + fi[i][j + 1] - Omega(x, y) \* pow(dx, 2)) / 4.0;

//Основная формула

fi\_tmp = (1 - gamma) \* fi[i][j] + gamma \* fi\_tmp;

delta = fi\_tmp - fi[i][j];

if (delta > delta\_max)

{

delta\_max = delta;

}

if (fi[i][j] > fi\_max)

{

fi\_max = fi[i][j];

}

fi[i][j] = fi\_tmp;

y = y + dx;

}

x = x + dx;

}

}

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> duration = end - start;

cout << "\nВремя вычислений: " << std::setprecision(8) << std::fixed << duration.count() << endl;

ofstream file("relaxation.txt");

x = 0;

file << "x,y,fi" << endl;

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

{

y = 0;

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

{

file << x << ',' << y << ',' << fi[i][j] << endl;

y = y + dx;

}

x = x + dx;

}

cout << "\nВычисления успешно загружены в файл: relaxation.txt\n" << endl;

}

void main()

{

setlocale(LC\_ALL, "Rus");

int N = int((x\_max - x\_min) / dx);

int M = int((y\_max - y\_min) / dx);

//Двумерный массив значений

double\*\* fi = new double\* [N + 1.0];

for (int i = 0; i < N + 1.0; i++)

{

fi[i] = new double[M + 1.0];

}

//Зануление массива

for (int i = 0; i < N + 1.0; i++)

{

for (int j = 0; j < M + 1.0; j++)

{

fi[i][j] = 0;

}

}

//Граничные условия

y = 0.0;

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

{

fi[N][j] = 2.0 \* pow(y, 2);

y = y + dx;

}

x = 0;

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

{

fi[i][M] = 6.0 \* pow(x, 2);

x = x + dx;

}

bool menu = true;

while (menu == true)

{

int input;

cout << "Выберите одно из действий:" << endl;

cout << "1 - вычисление по методу простых итераций" << endl;

cout << "2 - вычисление по методу последовательных смещений" << endl;

cout << "3 - вычисление по методу нижней релаксации" << endl;

cout << "4 - вычисление по методу верхней релаксации" << endl;

cout << "Другая кнопка - выход" << endl;

cin >> input;

if (input == 1)

Simple\_iter(fi, N, M);

else if (input == 2)

Successive\_disp(fi, N, M);

else if (input == 3)

Relaxation(0.5, fi, N, M);

else if (input == 4)

Relaxation(1.5, fi, N, M);

else

menu = false;

}

//Очистка памяти

for (int i = 0; i < N + 1.0; i++)

{

delete[] fi[i];

}

}

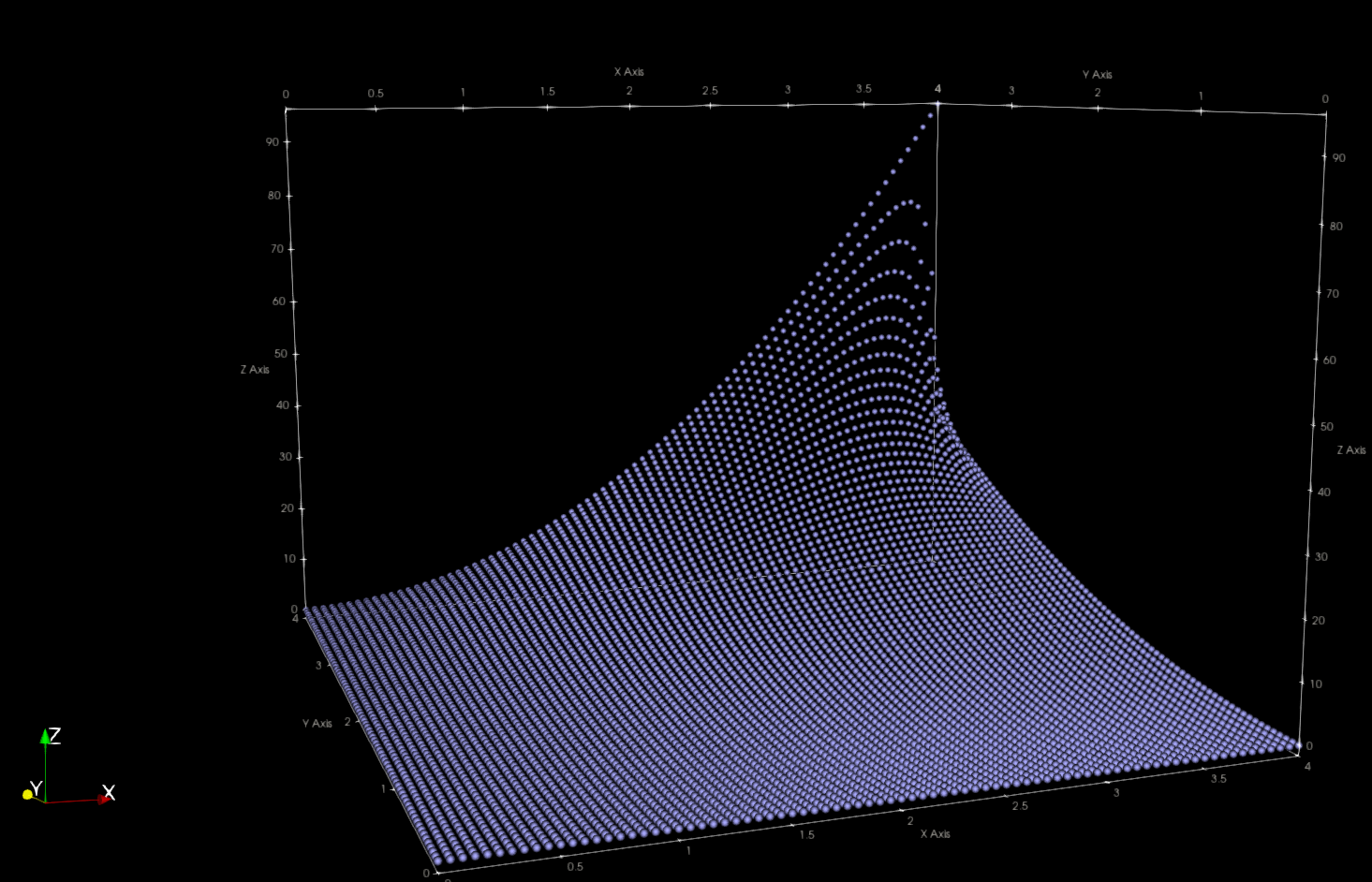
**Результаты.**

Сравнивались 3 итерационных метода решения: метод простых итераций, последовательных смещений и метод релаксации.

Метод простых итераций.



График.

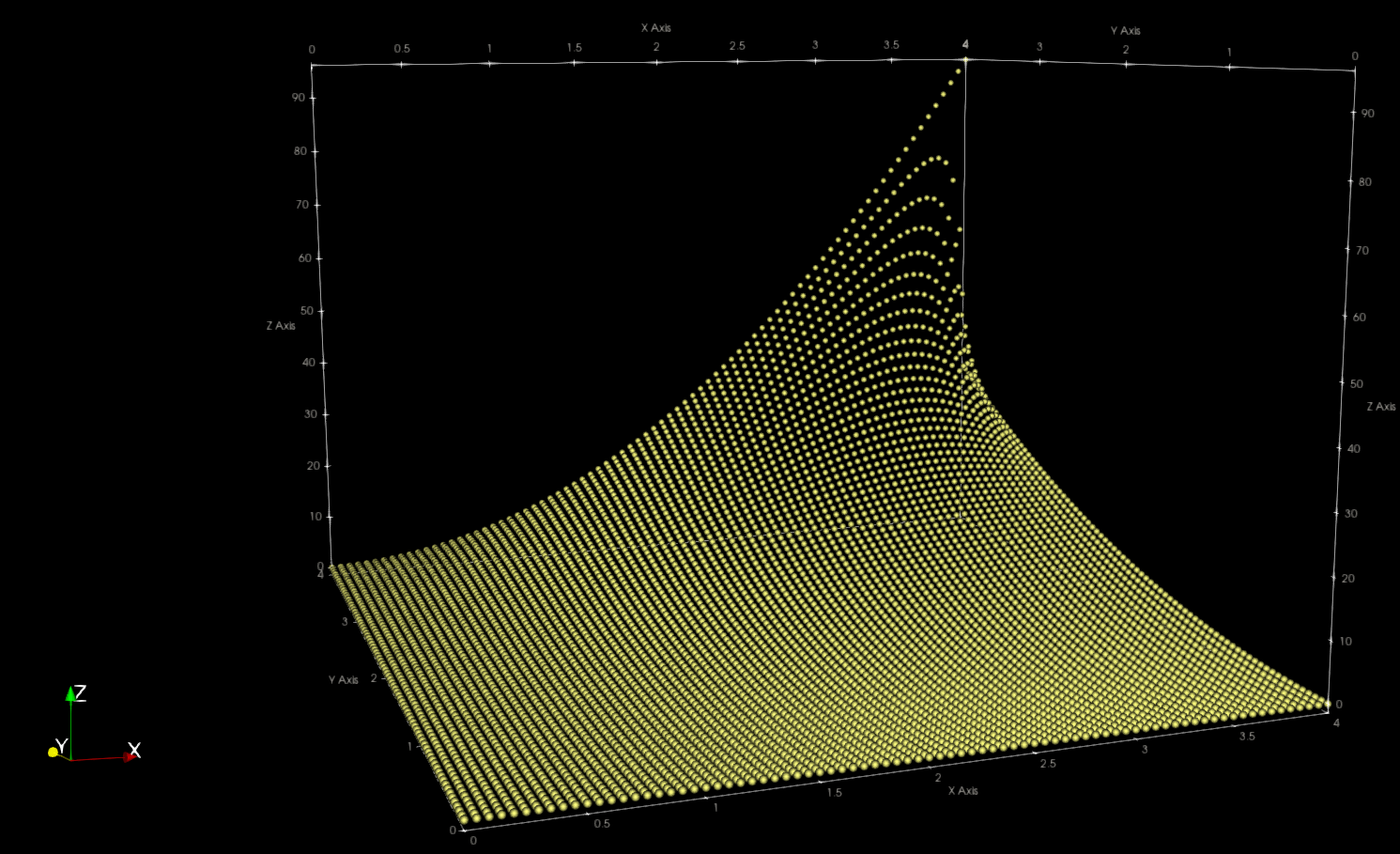


Метод последовательных смещений.

Данный метод значительно быстрее предыдущего.



График.



Метод релаксации.

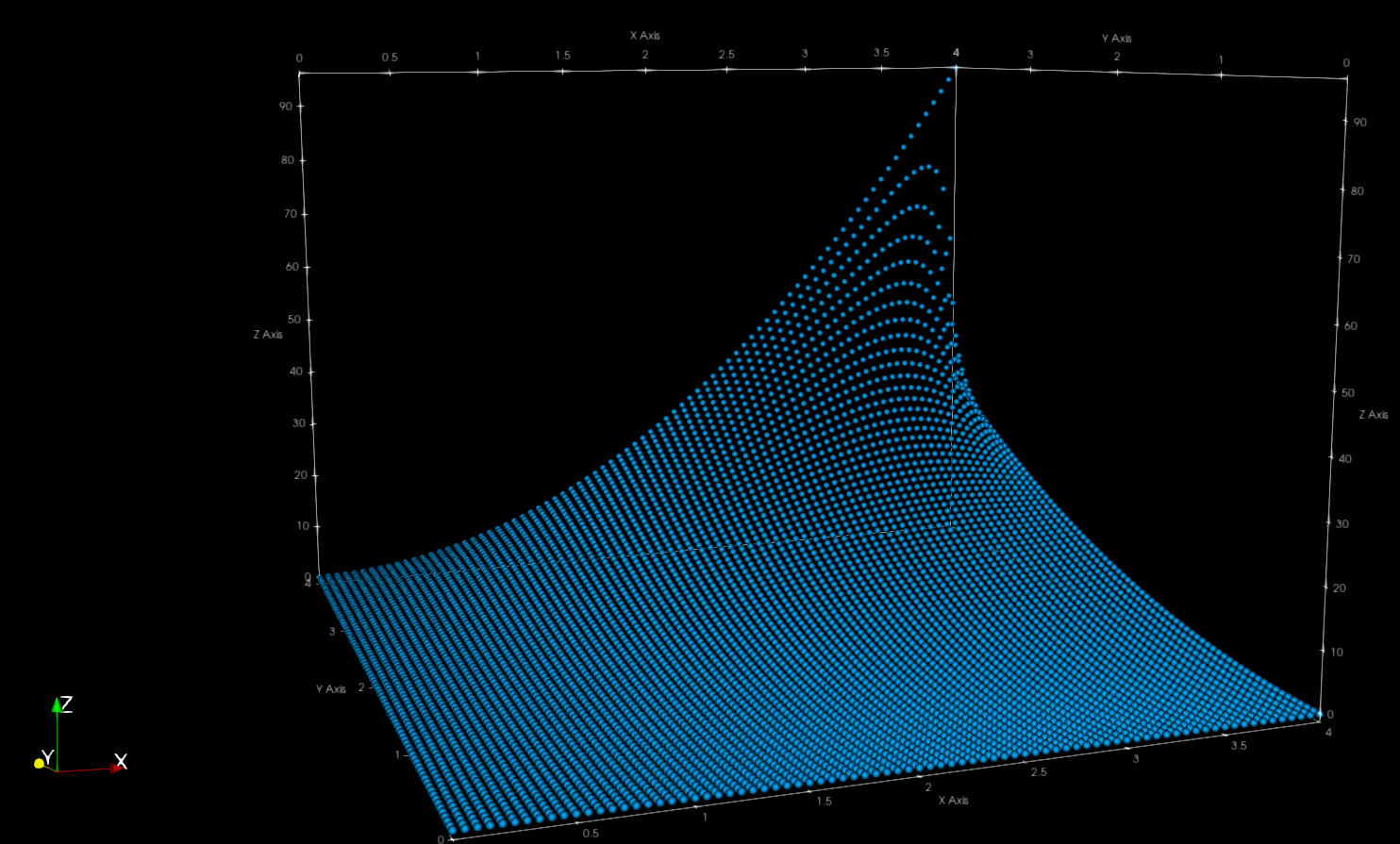
Метод нижней релаксации оказался медленней обоих предыдущих методов.



Метод верхней релаксации, наоборот, самый быстрый (в 2 раза быстрее последовательных смещений).



График метода верхней релаксации.



Все 3 метода на одном графике. Различия минимальны.

