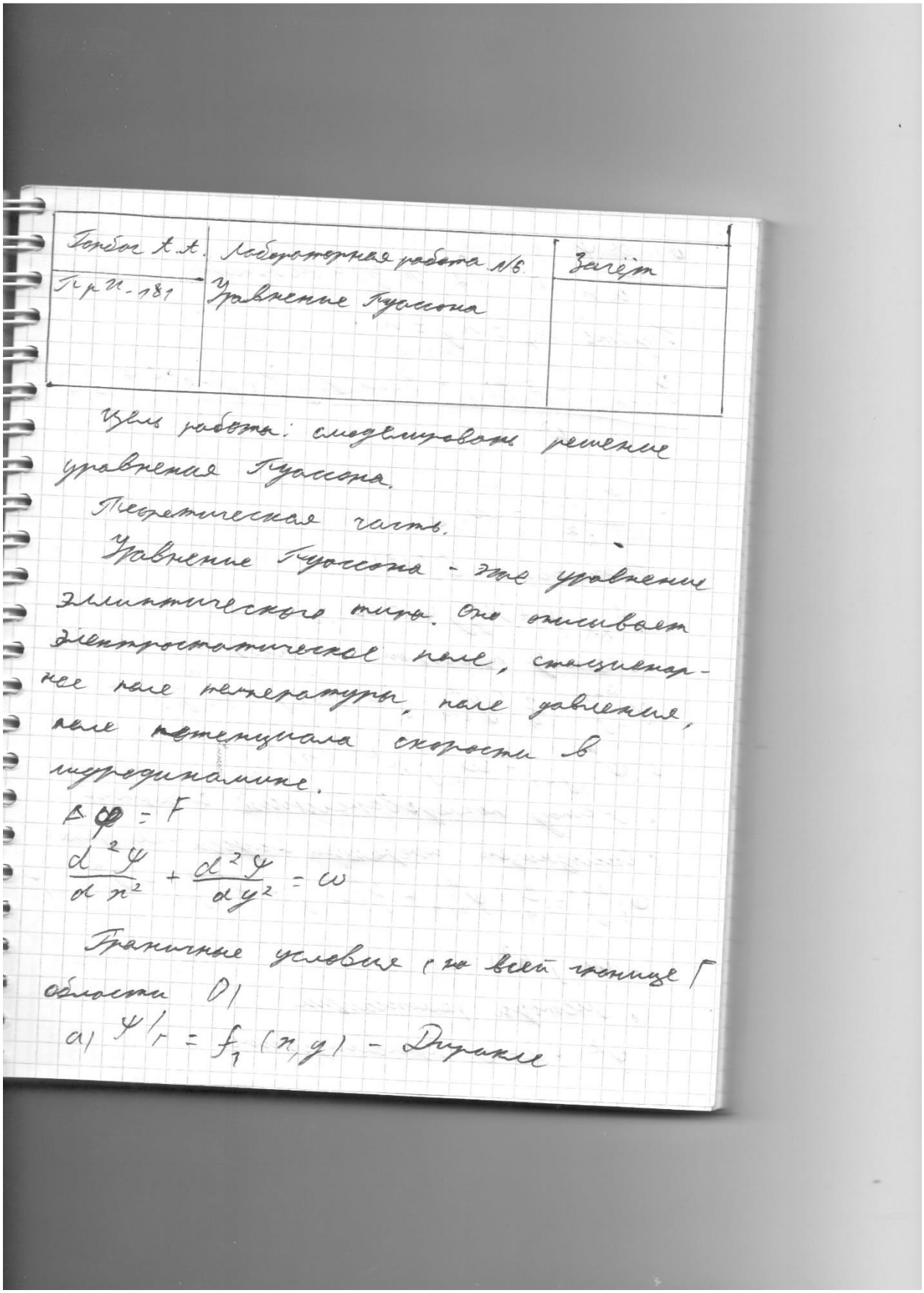
Лабораторная работа №6.

Конспект.



Голбан К.А.	Лабораторная работа №6	Зачёт
ГрИ-181	Уравнение Пуассона	

Цель работы: смоделировать решение уравнения Пуассона.

Теоретическая часть.

Уравнение Пуассона - это уравнение эллиптического типа. Оно описывает электростатическое поле, стационарное поле температуры, поле давления, поле потенциала скорости в гидродинамике.

$$\Delta \varphi = F$$

$$\frac{d^2\Psi}{dx^2} + \frac{d^2\Psi}{dy^2} = \omega$$

Граничные условия (на всей границе Γ области D)

а) $\Psi|_\Gamma = f_1(x,y)$ - Дирихле

б) $\frac{\partial \psi}{\partial n}\Big|_\Gamma = f_2(x,y)$ - Неймана

Пусть $h_x = h_y = h$;

$$\frac{\psi_{i-1,j} - 2\psi_{i,j} + \psi_{i+1,j}}{h^2} + \frac{\psi_{i,j-1} - 2\psi_{i,j} + \psi_{i,j+1}}{h^2} = w_{i,j}$$

Погрешность: $O(h^2)$

$$A_{xx}\psi + A_{yy}\psi = w_{i,j}$$

• метод простой итерации: (Якоби)

$$\psi_{i,j}^{k+1} = \frac{1}{4}(\psi_{i-1,j}^{k} + \psi_{i+1,j}^{k} + \psi_{i,j-1}^{k} + \psi_{i,j+1}^{k} - w_{i,j}h^2) \quad (1)$$

• метод последовательных смещений (с использованием итерации Гаусса - Зейделя)

$$\psi_{i,j}^{k+1} = \frac{1}{4}(\psi_{i-1}^{k+1} + \psi_{i+1,j}^{k} + \psi_{i,j-1}^{k+1} + \psi_{i,j+1}^{k} - w_{i,j}h^2) \quad (2)$$

• методы релаксации

ψ^k - k-е приближение

ψ^2 - уточнение по формуле (1) или (2)

$\psi_{i,j}^{k+1} = (1-\gamma)\psi_{i,j}^{k} + \gamma \psi_{i,j}^{2}$

$\gamma = 1 \Rightarrow$ обычный итерац. метод.

$0 < \gamma < 1$ - нижняя ~~релаксация~~ релаксация

$1 < \gamma < 2$ - верхняя релаксация.

Вывод: я изучил уравнение Гаусса и решил его тремя разными методами

Практическая часть.

Шаг dx = 0,5.

Граничные условия первого рода:

$u(0,y) = 0, u(4,y) = 2y^2$ и $u(x,0) = 0, u(x,4) = 6x^2$.

Рассчитываемая функция: $2x + y$.

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];
	}
	//Зануление 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];
	}
}
```

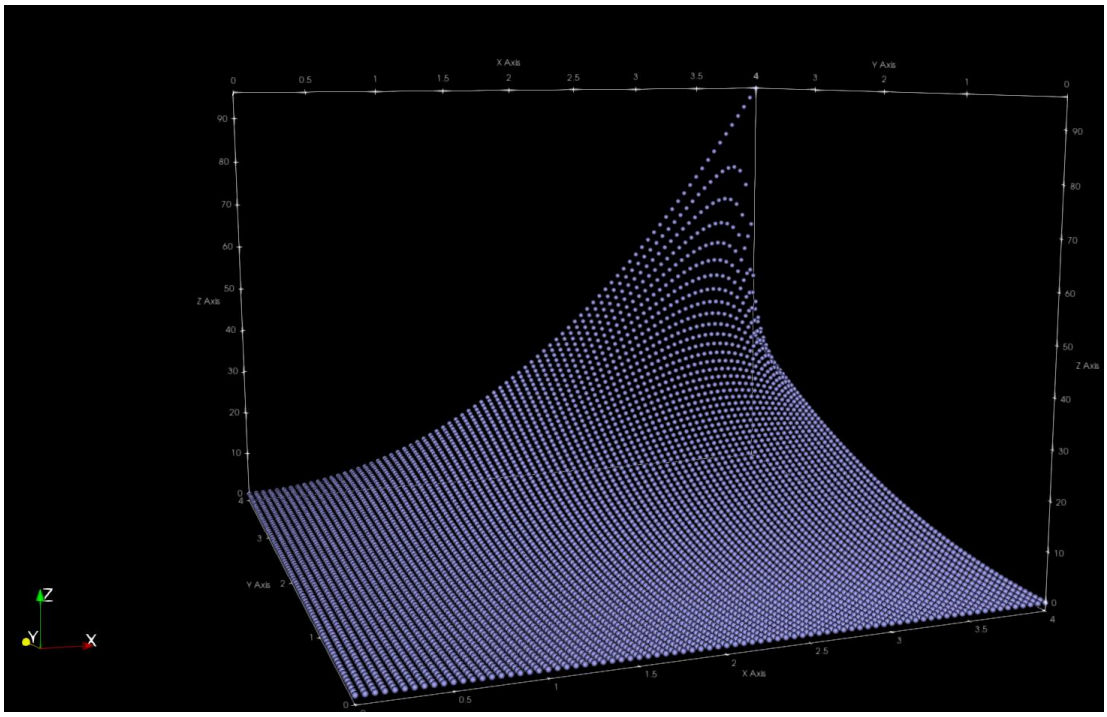
Результаты.

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

Метод простых итераций.

```
Время вычислений: 0.64856839

Вычисления успешно загружены в файл: simple.txt
```

График.

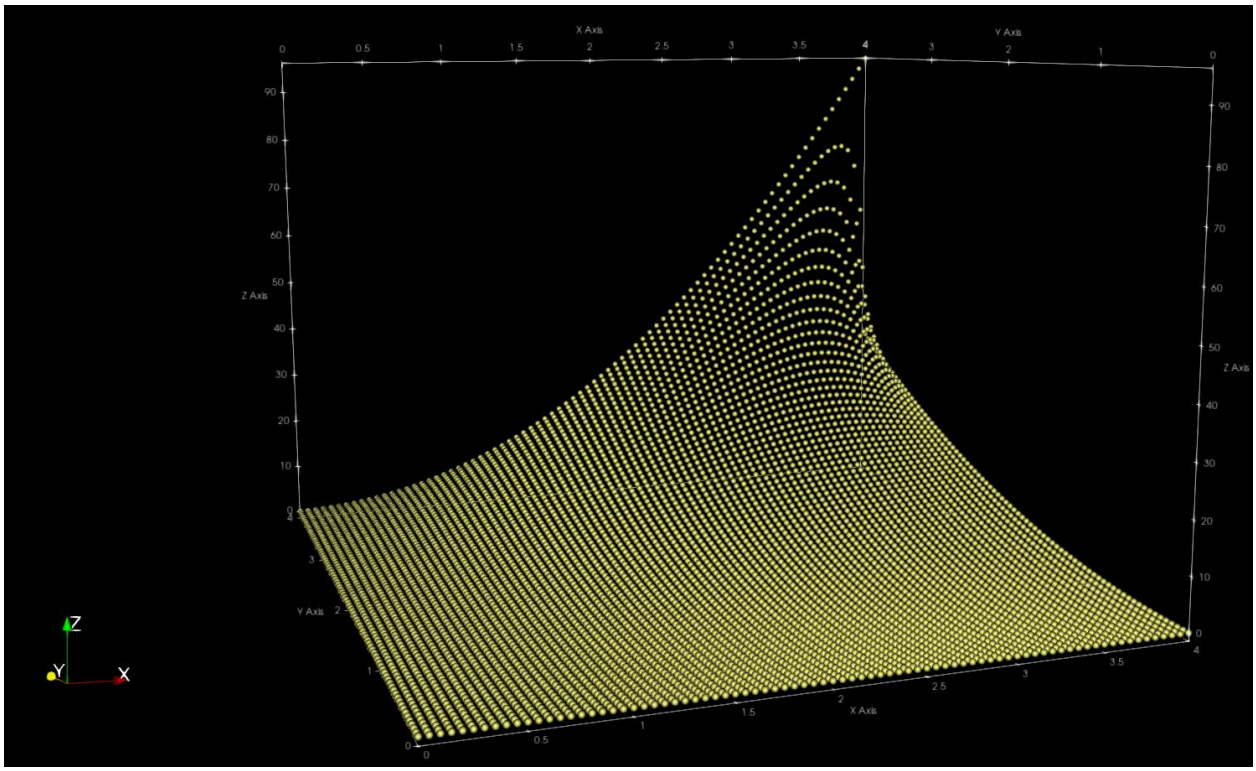


Метод последовательных смещений.

Данный метод значительно быстрее предыдущего.

```
Время вычислений: 0.41365370

Вычисления успешно загружены в файл: successive_disp.txt
```

График.



Метод релаксации.

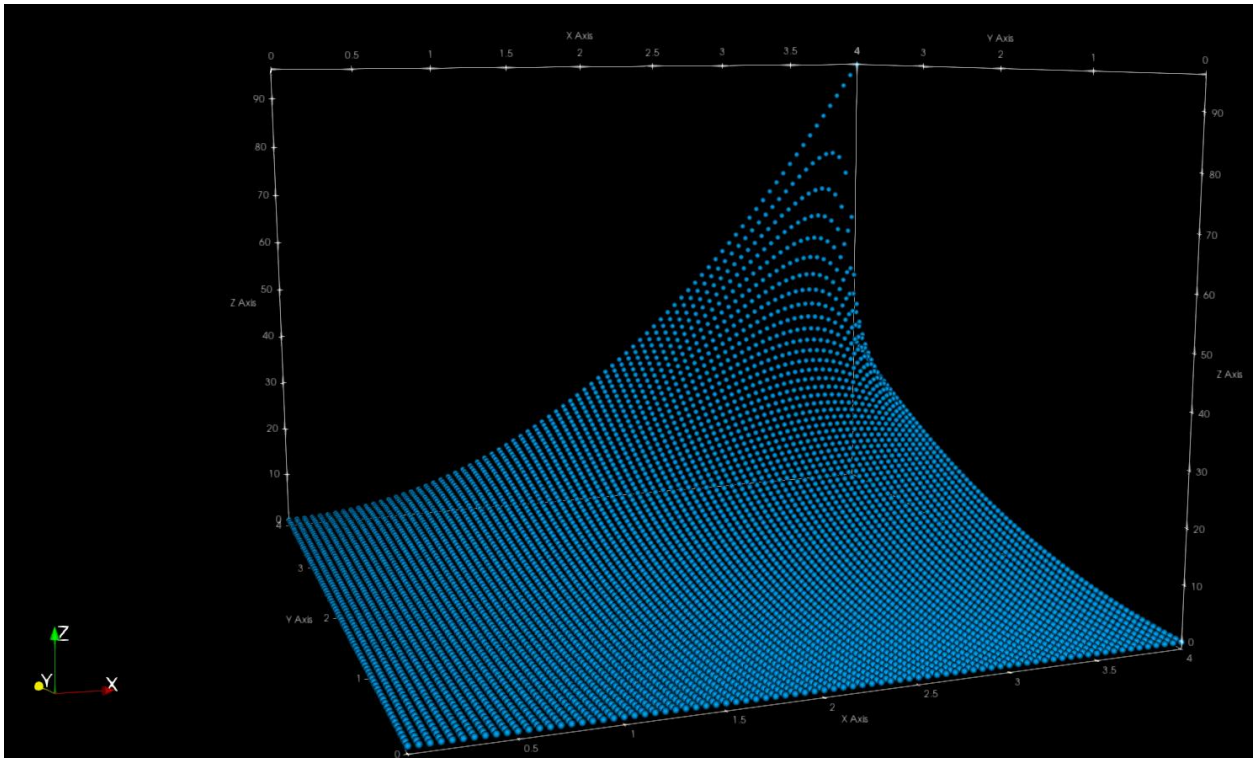
Метод нижней релаксации оказался медленней обоих предыдущих методов.

```
Время вычислений: 0.94625062

Вычисления успешно загружены в файл: relaxation.txt
```

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

```
Время вычислений: 0.21816850

Вычисления успешно загружены в файл: relaxation.txt
```

График метода верхней релаксации.



Все 3 метода на одном графике. Различия минимальны.