МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | **Факультет прикладной математики и информатики**

Кафедра прикладной математики

Лабораторная работа № 3

по дисциплине «КТМиАД»

ПОСТРОЕНИЕ ПОРТРЕТА И СБОРКА КОНЕЧНОЭЛЕМЕНТНОЙ МАТРИЦЫ

Бригада 10 БАРАНОВ ЯРОСЛАВ

Группа ПММ-32 МАКАРЫЧЕВ СЕРГЕЙ

Вариант 1

Преподаватели КОШКИНА Ю.И.

Новосибирск, 2023

1. Задание

Цель: реализовать алгоритм построения портрета и сборки конечноэлементной матрицы для разреженного строчного формата хранения, возникающие при решении эллиптической задачи. Протестировать написанную программу.

Вариант: лагранжевы базисные функции первого порядка.

2. Особенности реализации и возможности

Поддерживаются все три краевых условия. Для первого формируется массив узлов граничных граней, а для второго и третьего массивы треугольных граничных граней. В обоих случаях это происходит проходом по кубической сетке, но с некоторыми усложнениями во втором случае, такими как: разбиение грани на треугольники и вычисление номера элемента, которому принадлежит грань (для вычисления нормали и интерполянта).

Для более быстрого тестирования разработанной программы численно вычисляется оператор Лапласа. Также для более удобного тестирования и учета сложной геометрии расчетной области численно вычисляется производная по нормали, как векторное произведение единичной нормали и градиента (конечно, с учетом знака нормали, который определяется исходя из вершины тетраэдра, не лежащей на грани). Благодаря этому, краевые условия второго и третьего рода учитываются без ручного задания производной и для любого наклона грани.

Имеется возможность задания разрывных коэффициентов решаемого уравнения (λ, γ, β). Однако тестирование этой возможности требует специально составленного теста, к тому же для учёта краевых условий потребуется ручное вычисление функций, в отличие от неразрывных коэффициентов. Поэтому данная возможность тестироваться не будет.

В качестве решателя СЛАУ могут использоваться: BCGSTABLU, LOSLU, LU.

Матрица коэффициентов для L-координат вычисляется как обратная к матрице D, которая имеет вид:

$$D = \begin{pmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{pmatrix}$$

Алгоритм работает на базе метода Гаусса с постолбцовым выбором главного элемента, параллельно считается определитель матрицы D.

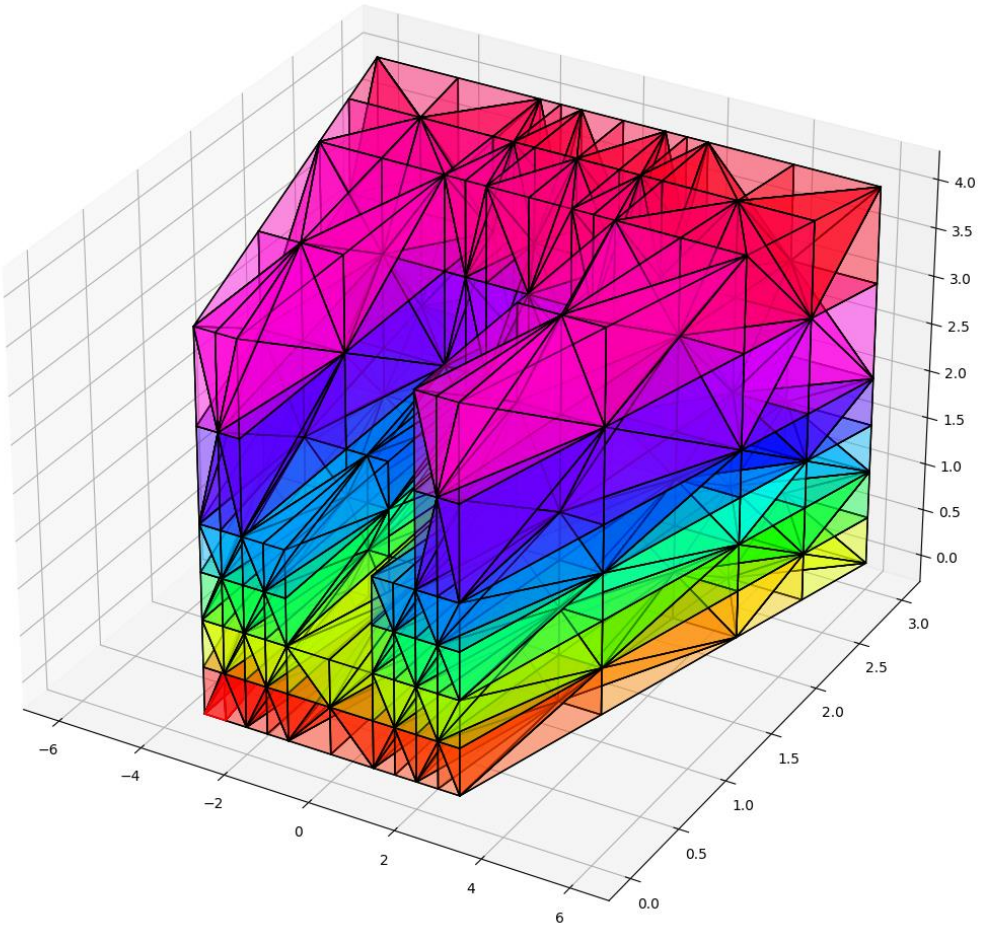
3. Тестирование

Паттерн разбиения куба: 5 тетраэдров. Решатель СЛАУ: BCGSTABLU.

Параметры задачи: $\lambda = 2, \gamma = 2, \beta = 2.$

На гранях параллельных плоскости Oyz заданы краевые условия первого рода (в данном случае имеются в виду ещё и две грани по бокам, которые имеют наибольшую по площади проекцию на эту плоскость), на гранях параллельных плоскости Oxz краевые условия второго рода, а на гранях параллельных плоскости Oxy краевые условия третьего рода.

3.1. Тестирование правильности работы программы на равномерной сетке



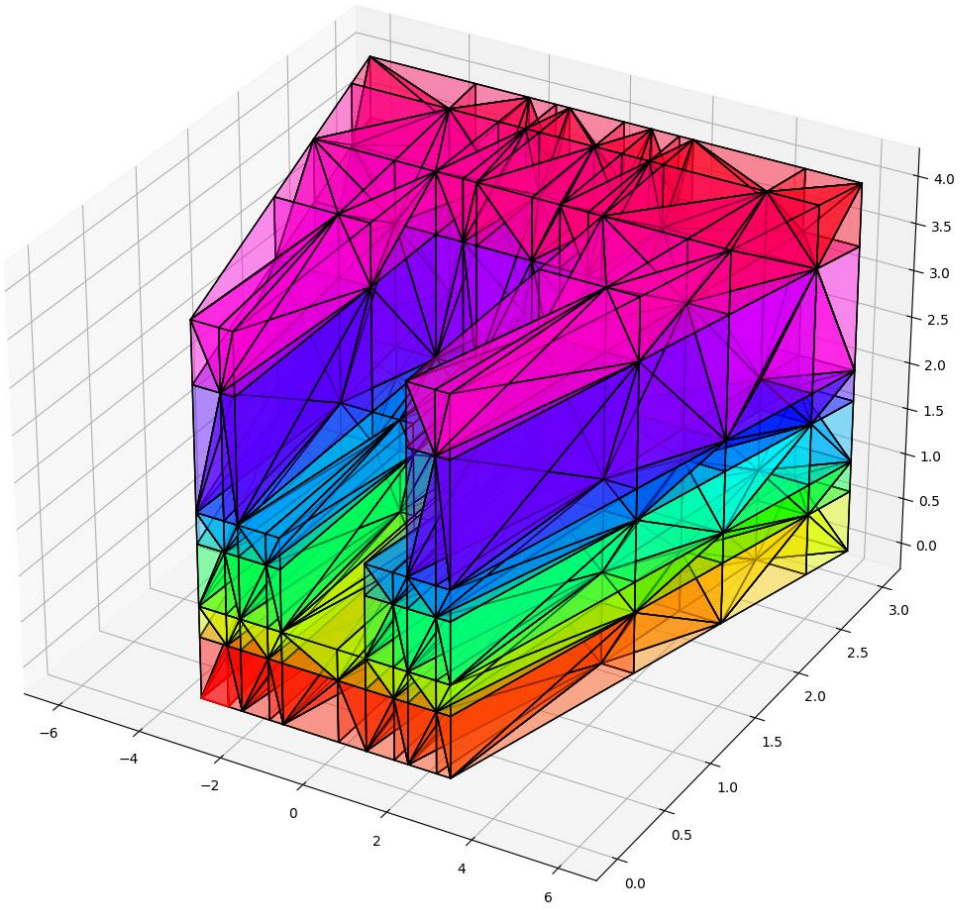
Тестируемая функция: $u = x + y + z$.

В центре таблицы имеются внутренние узлы.

X	Y	Z	u	u*	|u-u*|
-3.000	0.000	0.000	-3.000000000000000e+00	-3.000000000000000e+00	0.000000000000000e+00
-2.500	0.000	0.000	-2.500000000000000e+00	-2.500000000000000e+00	4.440892098500626e-16
-2.000	0.000	0.000	-1.999999999999999e+00	-2.000000000000000e+00	6.661338147750939e-16
-1.500	0.000	0.000	-1.500000000000000e+00	-1.500000000000000e+00	0.000000000000000e+00
...					
-4.000	1.000	0.500	-2.500000000000000e+00	-2.500000000000000e+00	4.440892098500626e-16
-3.000	1.000	0.500	-1.500000000000000e+00	-1.500000000000000e+00	4.440892098500626e-16
-2.000	1.000	0.500	-5.000000000000009e-01	-4.999999999999999e-01	9.992007221626409e-16
-1.500	1.000	0.500	-1.915134717478395e-15	0.000000000000000e+00	1.915134717478395e-15
-1.000	1.000	0.500	4.999999999999986e-01	5.000000000000000e-01	1.443289932012704e-15
0.000	1.000	0.500	1.499999999999999e+00	1.500000000000000e+00	8.881784197001252e-16
1.000	1.000	0.500	2.499999999999998e+00	2.500000000000000e+00	1.776356839400250e-15
1.500	1.000	0.500	3.000000000000000e+00	3.000000000000000e+00	4.440892098500626e-16
2.000	1.000	0.500	3.499999999999999e+00	3.500000000000000e+00	8.881784197001252e-16
3.000	1.000	0.500	4.500000000000000e+00	4.499999999999999e+00	8.881784197001252e-16
4.000	1.000	0.500	5.499999999999998e+00	5.499999999999999e+00	8.881784197001252e-16
-5.000	2.000	0.500	-2.500000000000000e+00	-2.499999999999999e+00	4.440892098500626e-16
-3.500	2.000	0.500	-9.999999999999999e-01	-9.999999999999993e-01	5.551115123125783e-16
-2.000	2.000	0.500	4.999999999999981e-01	5.000000000000000e-01	1.942890293094024e-15
-1.500	2.000	0.500	9.999999999999982e-01	1.000000000000000e+00	1.776356839400250e-15
-1.000	2.000	0.500	1.499999999999998e+00	1.500000000000000e+00	1.776356839400250e-15
0.000	2.000	0.500	2.500000000000002e+00	2.500000000000000e+00	1.776356839400250e-15
1.000	2.000	0.500	3.500000000000000e+00	3.500000000000000e+00	0.000000000000000e+00
1.500	2.000	0.500	4.000000000000001e+00	4.000000000000000e+00	8.881784197001252e-16
2.000	2.000	0.500	4.500000000000002e+00	4.500000000000000e+00	1.776356839400250e-15
3.500	2.000	0.500	6.000000000000000e+00	5.999999999999999e+00	8.881784197001252e-16
5.000	2.000	0.500	7.500000000000000e+00	7.499999999999999e+00	8.881784197001252e-16
-5.500	2.500	0.500	-2.500000000000000e+00	-2.500000000000000e+00	4.440892098500626e-16
-3.750	2.500	0.500	-7.500000000000007e-01	-7.500000000000000e-01	6.661338147750939e-16
-2.000	2.500	0.500	9.999999999999976e-01	1.000000000000000e+00	2.442490654175344e-15
-1.500	2.500	0.500	1.499999999999997e+00	1.500000000000000e+00	2.664535259100376e-15
-1.000	2.500	0.500	1.999999999999998e+00	2.000000000000000e+00	1.554312234475219e-15
0.000	2.500	0.500	3.000000000000001e+00	3.000000000000000e+00	8.881784197001252e-16
1.000	2.500	0.500	4.000000000000000e+00	4.000000000000000e+00	0.000000000000000e+00
1.500	2.500	0.500	4.500000000000000e+00	4.500000000000000e+00	0.000000000000000e+00
2.000	2.500	0.500	4.999999999999999e+00	5.000000000000000e+00	8.881784197001252e-16
3.750	2.500	0.500	6.750000000000003e+00	6.750000000000000e+00	2.664535259100376e-15
5.500	2.500	0.500	8.500000000000004e+00	8.500000000000000e+00	3.552713678800501e-15
...					
0.000	3.000	4.000	6.999999999999995e+00	7.000000000000000e+00	5.329070518200751e-15
1.000	3.000	4.000	7.999999999999996e+00	8.000000000000000e+00	4.440892098500626e-15
1.500	3.000	4.000	8.499999999999996e+00	8.500000000000000e+00	3.552713678800501e-15
2.000	3.000	4.000	8.999999999999996e+00	9.000000000000000e+00	3.552713678800501e-15
4.000	3.000	4.000	1.100000000000000e+01	1.100000000000000e+01	0.000000000000000e+00
6.000	3.000	4.000	1.300000000000000e+01	1.300000000000000e+01	1.776356839400250e-15
||u-u*||/||u*||					4.429713604077223e-16

3.2. Тестирование правильности работы программы на неравномерной сетке

Коэффициент разрядки: 0.5 для всех промежутков разбиения.



Тестируемая функция: $u = x + y + z$.

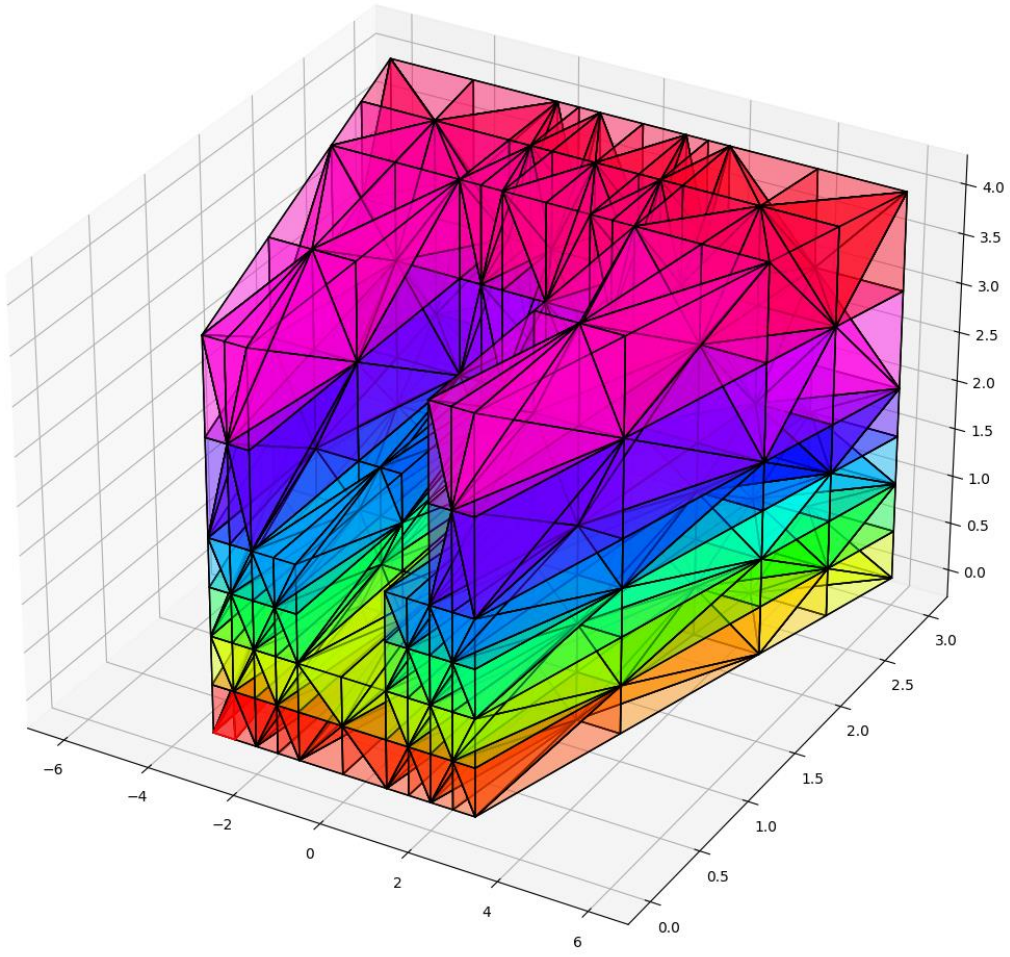
В центре таблицы имеются внутренние узлы.

X	Y	Z	u	u*	|u-u*|
-3.000	0.000	0.000	-3.000000000000001e+00	-3.000000000000000e+00	8.881784197001252e-16
-2.333	0.000	0.000	-2.333333333333333e+00	-2.333333333333333e+00	0.000000000000000e+00
-2.000	0.000	0.000	-2.000000000000001e+00	-2.000000000000000e+00	1.332267629550188e-15
-1.333	0.000	0.000	-1.333333333333334e+00	-1.333333333333333e+00	8.881784197001252e-16
...					
-4.333	1.333	0.667	-2.333333333333334e+00	-2.333333333333334e+00	0.000000000000000e+00
-2.778	1.333	0.667	-7.777777777777787e-01	-7.777777777777780e-01	6.661338147750939e-16
-2.000	1.333	0.667	2.775557561562891e-17	1.110223024625157e-16	8.326672684688674e-17
-1.333	1.333	0.667	6.666666666666656e-01	6.666666666666664e-01	7.771561172376096e-16
-1.000	1.333	0.667	9.999999999999993e-01	9.999999999999999e-01	5.551115123125783e-16
0.333	1.333	0.667	2.333333333333332e+00	2.333333333333333e+00	8.881784197001252e-16
1.000	1.333	0.667	2.999999999999998e+00	3.000000000000000e+00	1.776356839400250e-15
1.667	1.333	0.667	3.666666666666663e+00	3.666666666666667e+00	3.108624468950438e-15
2.000	1.333	0.667	3.999999999999994e+00	4.000000000000000e+00	5.329070518200751e-15
3.556	1.333	0.667	5.555555555555552e+00	5.555555555555555e+00	3.552713678800501e-15
4.333	1.333	0.667	6.333333333333336e+00	6.333333333333335e+00	8.881784197001252e-16
-5.000	2.000	0.667	-2.333333333333334e+00	-2.333333333333334e+00	0.000000000000000e+00
-3.000	2.000	0.667	-3.333333333333337e-01	-3.333333333333334e-01	3.330669073875470e-16
-2.000	2.000	0.667	6.666666666666669e-01	6.666666666666666e-01	2.220446049250313e-16
-1.333	2.000	0.667	1.333333333333333e+00	1.333333333333333e+00	4.440892098500626e-16
-1.000	2.000	0.667	1.666666666666666e+00	1.666666666666667e+00	4.440892098500626e-16
0.333	2.000	0.667	2.999999999999999e+00	3.000000000000000e+00	4.440892098500626e-16
1.000	2.000	0.667	3.666666666666666e+00	3.666666666666667e+00	4.440892098500626e-16
1.667	2.000	0.667	4.333333333333333e+00	4.333333333333333e+00	0.000000000000000e+00

```
|  2.000|  2.000|  0.667| 4.666666666666665e+00| 4.666666666666667e+00| 1.776356839400250e-15|
|  4.000|  2.000|  0.667| 6.666666666666663e+00| 6.666666666666666e+00| 3.552713678800501e-15|
|  5.000|  2.000|  0.667| 7.666666666666669e+00| 7.666666666666668e+00| 8.881784197001252e-16|
| -5.667|  2.667|  0.667|-2.333333333333333e+00|-2.333333333333333e+00| 0.000000000000000e+00|
| -3.222|  2.667|  0.667| 1.111111111111102e-01| 1.111111111111109e-01| 7.632783294297951e-16|
| -2.000|  2.667|  0.667| 1.333333333333335e+00| 1.333333333333333e+00| 1.332267629550188e-15|
| -1.333|  2.667|  0.667| 2.000000000000000e+00| 2.000000000000000e+00| 0.000000000000000e+00|
| -1.000|  2.667|  0.667| 2.333333333333333e+00| 2.333333333333333e+00| 4.440892098500626e-16|
|  0.333|  2.667|  0.667| 3.666666666666667e+00| 3.666666666666667e+00| 8.881784197001252e-16|
|  1.000|  2.667|  0.667| 4.333333333333332e+00| 4.333333333333333e+00| 8.881784197001252e-16|
|  1.667|  2.667|  0.667| 5.000000000000000e+00| 5.000000000000000e+00| 0.000000000000000e+00|
|  2.000|  2.667|  0.667| 5.333333333333330e+00| 5.333333333333333e+00| 2.664535259100376e-15|
|  4.444|  2.667|  0.667| 7.777777777777772e+00| 7.777777777777778e+00| 5.329070518200751e-15|
|  5.667|  2.667|  0.667| 9.000000000000000e+00| 8.999999999999998e+00| 1.776356839400250e-15|
...
|  1.000|  3.000|  4.000| 7.999999999999994e+00| 8.000000000000000e+00| 6.217248937900877e-15|
|  1.667|  3.000|  4.000| 8.666666666666659e+00| 8.666666666666666e+00| 7.105427357601002e-15|
|  2.000|  3.000|  4.000| 9.000000000000000e+00| 9.000000000000000e+00| 0.000000000000000e+00|
|  4.667|  3.000|  4.000| 1.166666666666665e+01| 1.166666666666667e+01| 1.421085471520200e-14|
|  6.000|  3.000|  4.000| 1.300000000000000e+01| 1.300000000000000e+01| 0.000000000000000e+00|
+-------------------------------------------------------------------------------------------+
|   ||u-u*||/||u*||                                                    5.943224582222025e-16|
+-------------------------------------------------------------------------------------------+
```

3.3. Определение порядка сходимости на равномерной сетке

Тестируемая функция: $u = \sin(x + y + z)$.



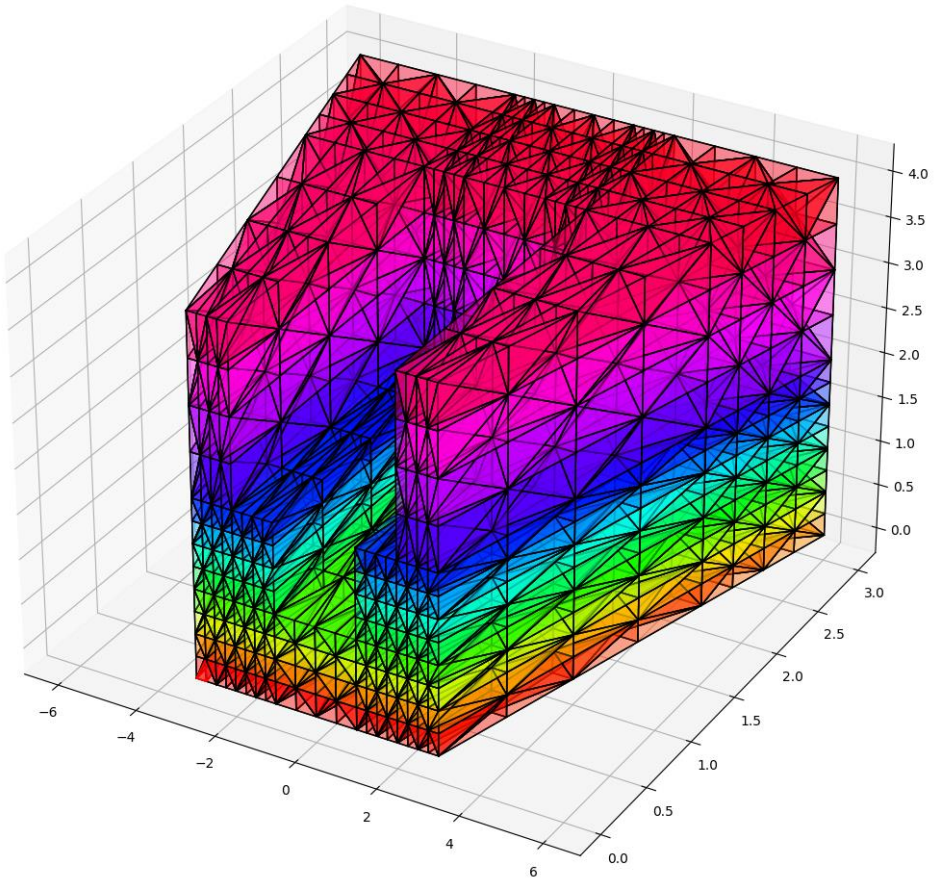
В центре таблицы имеются внутренние узлы.

X	Y	Z	u	u*	|u-u*|
-3.000	0.000	0.000	-1.411200080598672e-01	-1.411200080598672e-01	0.000000000000000e+00
-2.500	0.000	0.000	-6.073561632611917e-01	-5.984721441039564e-01	8.884019157235246e-03
-2.000	0.000	0.000	-9.216839816240435e-01	-9.092974268256817e-01	1.238655479836182e-02
-1.500	0.000	0.000	-9.061952645132352e-01	-9.974949866040544e-01	9.129972209081927e-02
...					
-4.000	1.000	0.500	-5.984721441039569e-01	-5.984721441039569e-01	0.000000000000000e+00
-3.000	1.000	0.500	-8.908594157939307e-01	-9.974949866040544e-01	1.066355708101238e-01
-2.000	1.000	0.500	-4.295541295983810e-01	-4.794255386042029e-01	4.987140900582193e-02
-1.500	1.000	0.500	3.607838648511591e-02	0.000000000000000e+00	3.607838648511591e-02
-1.000	1.000	0.500	5.142500098655611e-01	4.794255386042030e-01	3.482447126135813e-02
0.000	1.000	0.500	8.240566391643303e-01	9.974949866040544e-01	1.734383474397242e-01
1.000	1.000	0.500	5.557469665597314e-01	5.984721441039564e-01	4.272517754422500e-02
1.500	1.000	0.500	7.754927259203848e-02	1.411200080598672e-01	6.357073546782874e-02
2.000	1.000	0.500	-4.369726098832599e-01	-3.507832276896198e-01	8.618938219364003e-02
3.000	1.000	0.500	-6.642818666338209e-01	-9.775301176650969e-01	3.132482510312760e-01
4.000	1.000	0.500	-7.055403255703925e-01	-7.055403255703925e-01	0.000000000000000e+00
-5.000	2.000	0.500	-5.984721441039572e-01	-5.984721441039572e-01	0.000000000000000e+00
-3.500	2.000	0.500	-8.010956322515396e-01	-8.414709848078962e-01	4.037535255635660e-02
-2.000	2.000	0.500	3.310083820779590e-01	4.794255386042030e-01	1.484171565262440e-01
-1.500	2.000	0.500	7.396652783374382e-01	8.414709848078965e-01	1.018057064704583e-01
-1.000	2.000	0.500	8.546608902800377e-01	9.974949866040544e-01	1.428340963240168e-01
0.000	2.000	0.500	5.761779276527352e-01	5.984721441039564e-01	2.229421645122120e-02
1.000	2.000	0.500	-2.935281213533324e-01	-3.507832276896198e-01	5.725510633628744e-02
1.500	2.000	0.500	-6.754914007311783e-01	-7.568024953079282e-01	8.131109457674990e-02
2.000	2.000	0.500	-7.949251930881376e-01	-9.775301176650970e-01	1.826049245769594e-01
3.500	2.000	0.500	-1.982525103578547e-01	-2.794154981989267e-01	8.116298784107201e-02
5.000	2.000	0.500	9.379999767747385e-01	9.379999767747386e-01	1.110223024625157e-16
-5.500	2.500	0.500	-5.984721441039564e-01	-5.984721441039564e-01	0.000000000000000e+00
-3.750	2.500	0.500	-6.225609991197859e-01	-6.816387600233341e-01	5.907776090354822e-02
-2.000	2.500	0.500	6.603540887173612e-01	8.414709848078965e-01	1.811168960905353e-01
-1.500	2.500	0.500	8.357379822339577e-01	9.974949866040544e-01	1.617570043700968e-01
-1.000	2.500	0.500	8.321955586316130e-01	9.092974268256817e-01	7.710186819406872e-02
0.000	2.500	0.500	1.401683842444131e-01	1.411200080598672e-01	9.516238154541334e-04
1.000	2.500	0.500	-6.866688266000690e-01	-7.568024953079282e-01	7.013366870785920e-02
1.500	2.500	0.500	-8.409871740047852e-01	-9.775301176650970e-01	1.365429436603118e-01
2.000	2.500	0.500	-8.334824413747630e-01	-9.589242746631385e-01	1.254418332883754e-01
3.750	2.500	0.500	2.776443011972997e-01	4.500440737806176e-01	1.723997725833179e-01
5.500	2.500	0.500	7.984871126234904e-01	7.984871126234903e-01	1.110223024625157e-16
...					
1.000	3.000	4.000	9.979923877078725e-01	9.893582466233818e-01	8.634141084490676e-03
1.500	3.000	4.000	7.337088933872897e-01	7.984871126234903e-01	6.477821923620053e-02
2.000	3.000	4.000	3.039324381635013e-01	4.121184852417566e-01	1.081860470782553e-01
4.000	3.000	4.000	-6.836375586647730e-01	-9.999902065507035e-01	3.163526478859304e-01
6.000	3.000	4.000	4.201670368266408e-01	4.201670368266409e-01	1.110223024625157e-16

||u-u*||/||u*||	1.632441308846665e-01

Для вложенной сетки приведём лишь расчетную сетку и отношение погрешностей (в виду размера).

||u-u*||/||u*||	5.028554804157395e-02



Для дважды вложенной сетки приведём лишь отношение погрешностей:

```
+-----------------------------------------------------------------------------------------+
|   ||u-u*||/||u*||                                                 1.722648991871148e-02|
+-----------------------------------------------------------------------------------------+
```

Определим порядок сходимости:

$$\frac{\left|\left|u^* - u^h\right|\right|}{\left|\left|u^*\right|\right|} = 1.6324413088466 65\text{e} - 01$$

$$\frac{\left|\left|u^* - u^{\frac{h}{2}}\right|\right|}{\left|\left|u^*\right|\right|} = 5.028554804157395\text{e} - 02$$

$$\frac{\left|\left|u^* - u^{\frac{h}{4}}\right|\right|}{\left|\left|u^*\right|\right|} = 1.722648991871148\text{e} - 02$$

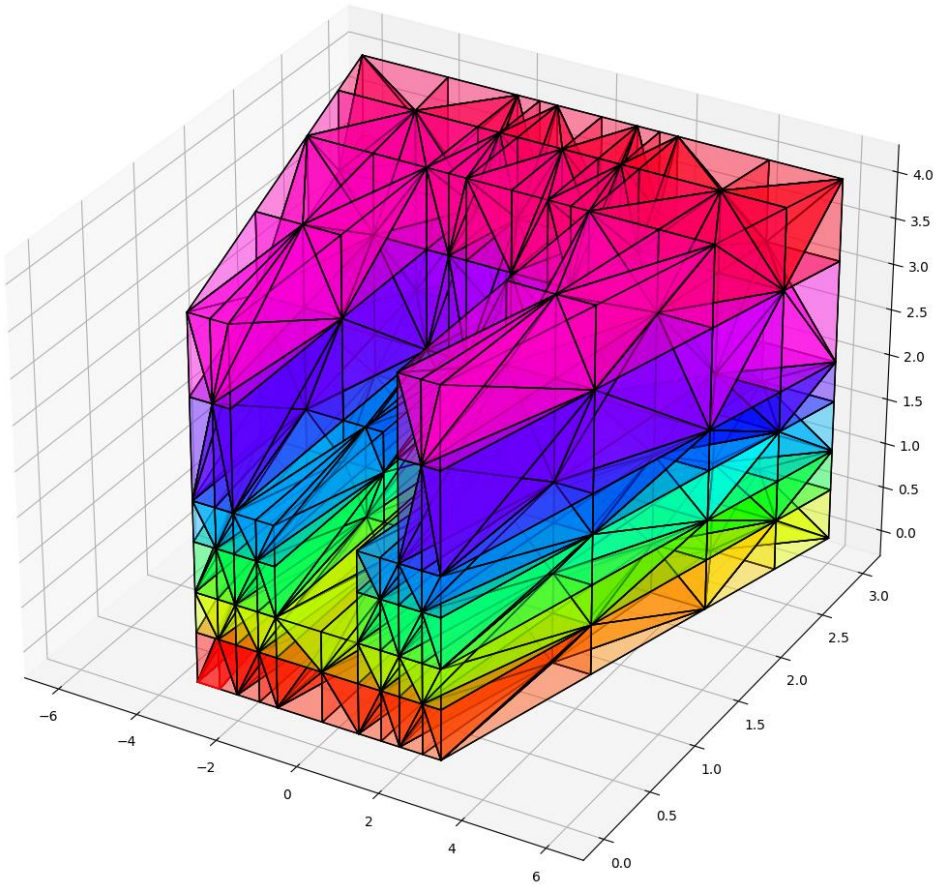
$$\log_2 \frac{\left|\left|u^* - u^h\right|\right|}{\left|\left|u^* - u^{\frac{h}{2}}\right|\right|} \approx 1.69$$

$$\log_2 \frac{\left\| u^* - u^{\frac{h}{2}} \right\|}{\left\| u^* - u^{\frac{h}{4}} \right\|} \approx 1.54$$

3.4. Определение порядка сходимости на неравномерной сетке

Тестируемая функция: $u = \sin(x + y + z)$.

Коэффициент разрядки: 0.8 для всех промежутков разбиения.



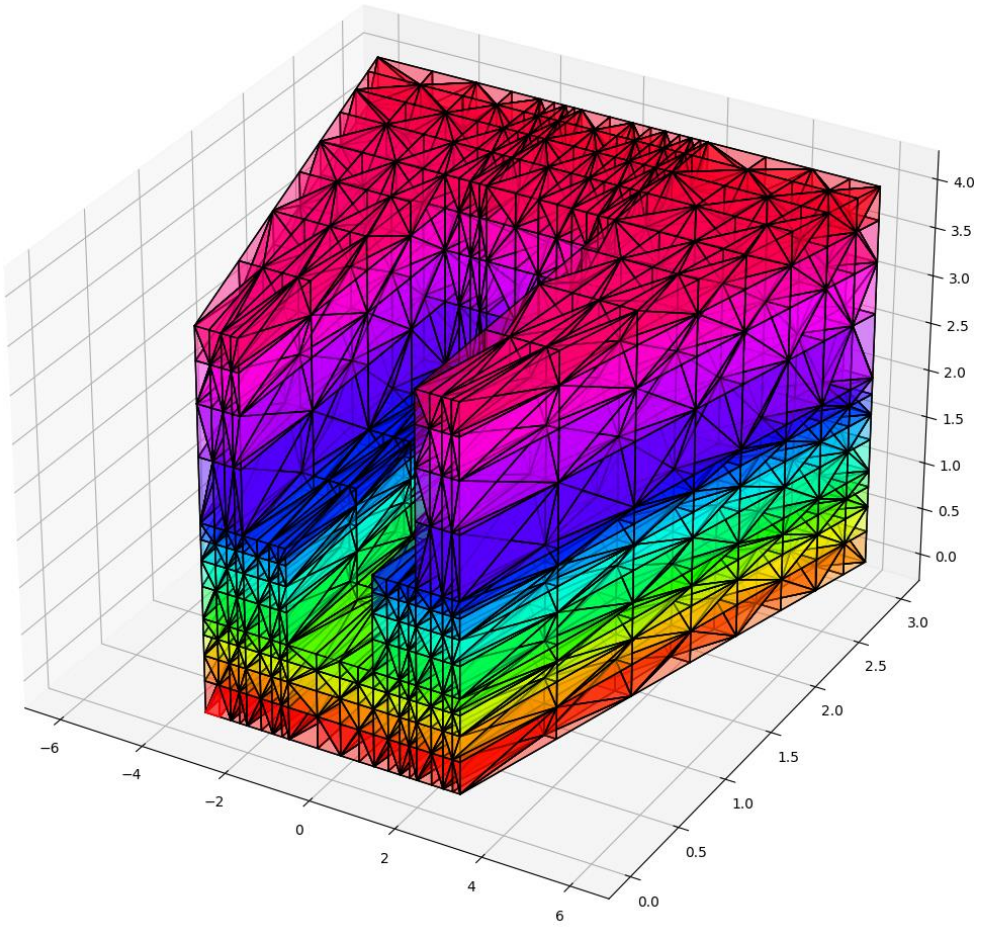
В центре таблицы имеются внутренние узлы.

X	Y	Z	u	u*	|u-u*|
-3.000	0.000	0.000	-1.411200080598672e-01	-1.411200080598672e-01	0.000000000000000e+00
-2.444	0.000	0.000	-6.514945547139539e-01	-6.420339006228256e-01	9.460654091128284e-03
-2.000	0.000	0.000	-9.520024580962810e-01	-9.092974268256817e-01	4.270503127059933e-02
-1.444	0.000	0.000	-8.759292894301988e-01	-9.920282150459668e-01	1.160989256157681e-01
...					
-4.111	1.111	0.556	-6.420339006228264e-01	-6.420339006228263e-01	1.110223024625157e-16
-2.938	1.111	0.556	-8.395969626518346e-01	-9.555751378057021e-01	1.159781751538675e-01
-2.000	1.111	0.556	-3.120882374031673e-01	-3.271946967961522e-01	1.510645939298483e-02
-1.444	1.111	0.556	2.128576308301539e-01	2.203977434561224e-01	7.540112625968431e-03
-1.000	1.111	0.556	6.164580769458687e-01	6.183698030697371e-01	1.911726123868362e-03
0.111	1.111	0.556	8.059475161615074e-01	9.786557044658371e-01	1.727081883043297e-01

```
|  1.000|  1.111|  0.556| 4.490411460945353e-01| 4.572726266358116e-01| 8.231480541276315e-03|
|  1.556|  1.111|  0.556|-9.542107104102096e-02|-8.054223317739648e-02| 1.487883786362448e-02|
|  2.000|  1.111|  0.556|-5.345521527956546e-01|-5.012770485883451e-01| 3.327510420730950e-02|
|  3.173|  1.111|  0.556|-6.065597446773525e-01|-9.919314832575790e-01| 3.853717385802264e-01|
|  4.111|  1.111|  0.556|-4.841640594687089e-01|-4.841640594687089e-01| 0.000000000000000e+00|
| -5.000|  2.000|  0.556|-6.420339006228257e-01|-6.420339006228256e-01| 1.110223024625157e-16|
| -3.333|  2.000|  0.556|-6.714135922279031e-01|-7.016978761467351e-01| 3.028428391883198e-02|
| -2.000|  2.000|  0.556| 3.883244754467167e-01| 5.274153857718655e-01| 1.390909103251489e-01|
| -1.444|  2.000|  0.556| 8.132312484949954e-01| 8.961922010299563e-01| 8.296095253496094e-02|
| -1.000|  2.000|  0.556| 8.706027730473229e-01| 9.998838616941024e-01| 1.292810886467795e-01|
|  0.111|  2.000|  0.556| 4.731114945060174e-01| 4.572726266358116e-01| 1.583886787020578e-02|
|  1.000|  2.000|  0.556|-3.402636474763147e-01|-4.022406483887265e-01| 6.197700091241187e-02|
|  1.556|  2.000|  0.556|-7.445277060091334e-01|-8.246134019729868e-01| 8.008569596385340e-02|
|  2.000|  2.000|  0.556|-8.050009256786728e-01|-9.877268260521109e-01| 1.827259003734382e-01|
|  3.667|  2.000|  0.556|-5.490006988444284e-02|-6.092533044599045e-02| 6.025260561547605e-03|
|  5.000|  2.000|  0.556| 9.558004312710241e-01| 9.558004312710241e-01| 0.000000000000000e+00|
| -5.556|  2.556|  0.556|-6.420339006228257e-01|-6.420339006228256e-01| 1.110223024625157e-16|
| -3.580|  2.556|  0.556|-4.230037438140585e-01|-4.521156252455272e-01| 2.911188143146876e-02|
| -2.000|  2.556|  0.556| 7.376413260122389e-01| 8.961922010299562e-01| 1.585508750177174e-01|
| -1.444|  2.556|  0.556| 8.514743118007795e-01| 9.954079577517649e-01| 1.439336459509855e-01|
| -1.000|  2.556|  0.556| 7.821039633237111e-01| 8.575467969251048e-01| 7.544283360139370e-02|
|  0.111|  2.556|  0.556|-3.092863846240319e-02|-8.054223317739648e-02| 4.961359471499328e-02|
|  1.000|  2.556|  0.556|-7.524046021464829e-01|-8.246134019729868e-01| 7.220879982650397e-02|
|  1.556|  2.556|  0.556|-8.589240558150856e-01|-9.989549170979283e-01| 1.400308612828427e-01|
|  2.000|  2.556|  0.556|-7.869800631427873e-01|-9.215578675785389e-01| 1.345778044357515e-01|
|  3.975|  2.556|  0.556| 4.175532088619540e-01| 7.196057947772551e-01| 3.020525859153011e-01|
|  5.556|  2.556|  0.556| 6.875512151130616e-01| 6.875512151130616e-01| 0.000000000000000e+00|
...
|  1.000|  3.000|  4.000| 9.886810394212360e-01| 9.893582466233818e-01| 6.772072021458131e-04|
|  1.556|  3.000|  4.000| 7.168748974123893e-01| 7.638272922377841e-01| 4.695239482539482e-02|
|  2.000|  3.000|  4.000| 3.108946977689528e-01| 4.121184852417566e-01| 1.012237874728038e-01|
|  4.222|  3.000|  4.000|-6.926142331383961e-01|-9.744251189104484e-01| 2.818108857720523e-01|
|  6.000|  3.000|  4.000| 4.201670368266409e-01| 4.201670368266409e-01| 5.551115123125783e-17|
+-------------------------------------------------------------------------------------------+
|   ||u-u*||/||u*||                                                    1.731342623726609e-01|
+-------------------------------------------------------------------------------------------+
```

Для вложенной сетки приведём лишь расчетную сетку и отношение погрешностей (в виду размера).

```
+-------------------------------------------------------------------------------------------+
|   ||u-u*||/||u*||                                                    6.806463083834428e-02|
+-------------------------------------------------------------------------------------------+
```



Для дважды вложенной сетки приведём лишь отношение погрешностей:

||u-u*||/||u*||	4.809735376973825e-02

Определим порядок сходимости:

$$\frac{\left|\left|u^* - u^h\right|\right|}{\left|\left|u^*\right|\right|} = 1.7313426237266 09\text{e} - 01$$

$$\frac{\left|\left|u^* - u^{\frac{h}{2}}\right|\right|}{\left|\left|u^*\right|\right|} = 6.8064630838344 28\text{e} - 02$$

$$\frac{\left|\left|u^* - u^{\frac{h}{4}}\right|\right|}{\left|\left|u^*\right|\right|} = 4.8097353769738 25\text{e} - 02$$

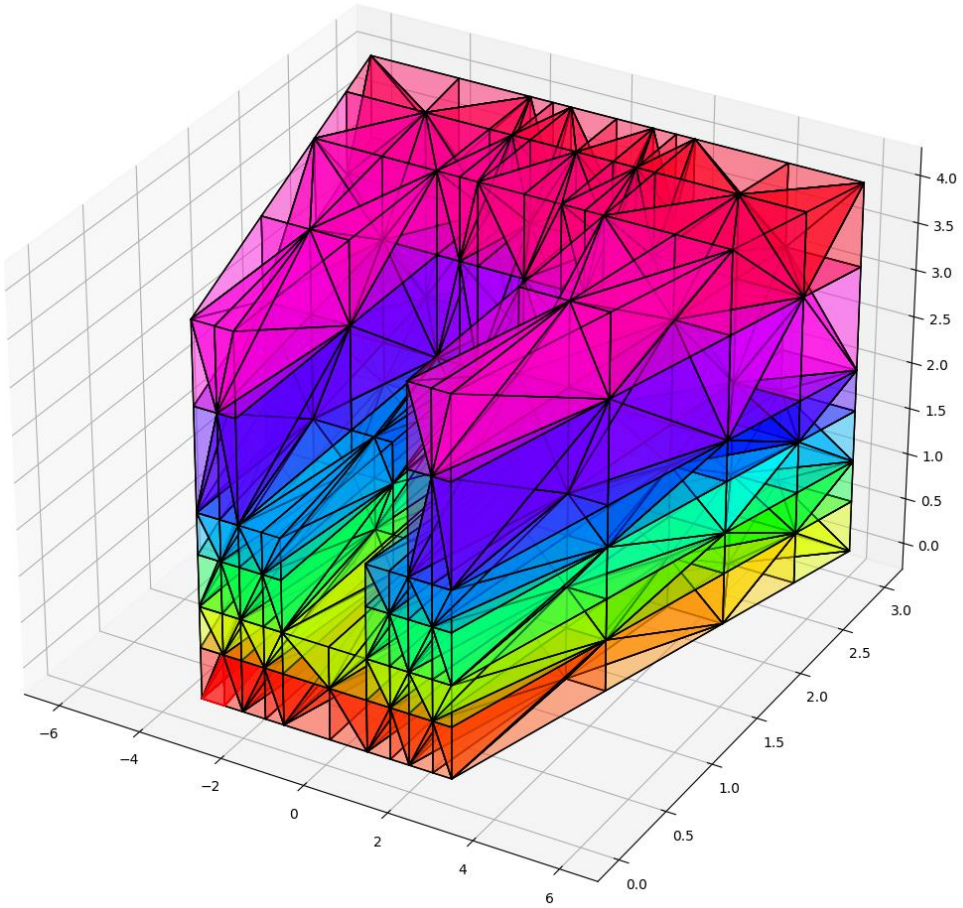
$$\log_2 \frac{\left|\left|u^* - u^h\right|\right|}{\left|\left|u^* - u^{\frac{h}{2}}\right|\right|} \approx 1.34$$

$$\log_2 \frac{\left\| u^* - u^{\frac{h}{2}} \right\|}{\left\| u^* - u^{\frac{h}{4}} \right\|} \approx 0.5$$

3.5. Тест, где на непараллельных координатным плоскостям гранях задано первое краевое условие, а на всех остальных второе.

Тестируемая функция: $u = 2y$.

Коэффициент разрядки: 0.8 для всех промежутков разбиения.



В центре таблицы имеются внутренние узлы.

X	Y	Z	u	u*	|u-u*|
-3.000	0.000	0.000	0.000000000000000e+00	0.000000000000000e+00	0.000000000000000e+00
-2.444	0.000	0.000	-4.850968053490435e-16	0.000000000000000e+00	4.850968053490435e-16
-2.000	0.000	0.000	-8.519142806224574e-16	0.000000000000000e+00	8.519142806224574e-16
-1.444	0.000	0.000	-7.763497914773479e-16	0.000000000000000e+00	7.763497914773479e-16
...					
-4.111	1.111	0.556	2.222222222222223e+00	2.222222222222222e+00	4.440892098500626e-16

```
| -2.938|  1.111|  0.556| 2.222222222222221e+00| 2.222222222222222e+00| 1.776356839400250e-15|
| -2.000|  1.111|  0.556| 2.222222222222221e+00| 2.222222222222222e+00| 1.776356839400250e-15|
| -1.444|  1.111|  0.556| 2.222222222222221e+00| 2.222222222222222e+00| 1.776356839400250e-15|
| -1.000|  1.111|  0.556| 2.222222222222221e+00| 2.222222222222222e+00| 8.881784197001252e-16|
|  0.111|  1.111|  0.556| 2.222222222222222e+00| 2.222222222222222e+00| 0.000000000000000e+00|
|  1.000|  1.111|  0.556| 2.222222222222222e+00| 2.222222222222222e+00| 0.000000000000000e+00|
|  1.556|  1.111|  0.556| 2.222222222222224e+00| 2.222222222222222e+00| 1.776356839400250e-15|
|  2.000|  1.111|  0.556| 2.222222222222224e+00| 2.222222222222222e+00| 1.776356839400250e-15|
|  3.173|  1.111|  0.556| 2.222222222222225e+00| 2.222222222222222e+00| 2.220446049250313e-15|
|  4.111|  1.111|  0.556| 2.222222222222223e+00| 2.222222222222222e+00| 4.440892098500626e-16|
| -5.000|  2.000|  0.556| 4.000000000000000e+00| 4.000000000000000e+00| 0.000000000000000e+00|
| -3.333|  2.000|  0.556| 3.999999999999995e+00| 4.000000000000000e+00| 4.884981308350689e-15|
| -2.000|  2.000|  0.556| 3.999999999999996e+00| 4.000000000000000e+00| 3.552713678800501e-15|
| -1.444|  2.000|  0.556| 3.999999999999996e+00| 4.000000000000000e+00| 4.440892098500626e-15|
| -1.000|  2.000|  0.556| 3.999999999999997e+00| 4.000000000000000e+00| 3.108624468950438e-15|
|  0.111|  2.000|  0.556| 4.000000000000003e+00| 4.000000000000000e+00| 2.664535259100376e-15|
|  1.000|  2.000|  0.556| 4.000000000000000e+00| 4.000000000000000e+00| 4.440892098500626e-16|
|  1.556|  2.000|  0.556| 3.999999999999999e+00| 4.000000000000000e+00| 1.332267629550188e-15|
|  2.000|  2.000|  0.556| 3.999999999999999e+00| 4.000000000000000e+00| 8.881784197001252e-16|
|  3.667|  2.000|  0.556| 4.000000000000000e+00| 4.000000000000000e+00| 0.000000000000000e+00|
|  5.000|  2.000|  0.556| 4.000000000000000e+00| 4.000000000000000e+00| 0.000000000000000e+00|
| -5.556|  2.556|  0.556| 5.111111111111110e+00| 5.111111111111111e+00| 8.881784197001252e-16|
| -3.580|  2.556|  0.556| 5.111111111111104e+00| 5.111111111111111e+00| 6.217248937900877e-15|
| -2.000|  2.556|  0.556| 5.111111111111107e+00| 5.111111111111111e+00| 3.552713678800501e-15|
| -1.444|  2.556|  0.556| 5.111111111111108e+00| 5.111111111111111e+00| 2.664535259100376e-15|
| -1.000|  2.556|  0.556| 5.111111111111108e+00| 5.111111111111111e+00| 2.664535259100376e-15|
|  0.111|  2.556|  0.556| 5.111111111111107e+00| 5.111111111111111e+00| 3.552713678800501e-15|
|  1.000|  2.556|  0.556| 5.111111111111109e+00| 5.111111111111111e+00| 1.776356839400250e-15|
|  1.556|  2.556|  0.556| 5.111111111111112e+00| 5.111111111111111e+00| 8.881784197001252e-16|
|  2.000|  2.556|  0.556| 5.111111111111109e+00| 5.111111111111111e+00| 1.776356839400250e-15|
|  3.975|  2.556|  0.556| 5.111111111111107e+00| 5.111111111111111e+00| 3.552713678800501e-15|
...
|  1.000|  3.000|  4.000| 5.999999999999996e+00| 6.000000000000000e+00| 3.552713678800501e-15|
|  1.556|  3.000|  4.000| 5.999999999999996e+00| 6.000000000000000e+00| 3.552713678800501e-15|
|  2.000|  3.000|  4.000| 5.999999999999997e+00| 6.000000000000000e+00| 2.664535259100376e-15|
|  4.222|  3.000|  4.000| 6.000000000000001e+00| 6.000000000000000e+00| 8.881784197001252e-16|
|  6.000|  3.000|  4.000| 6.000000000000000e+00| 6.000000000000000e+00| 0.000000000000000e+00|
+-------------------------------------------------------------------------------------------+
|   ||u-u*||/||u*||                                                      6.583655059539784e-16|
+-------------------------------------------------------------------------------------------+
```

4. Выводы

Анализируя результаты тестирования, можно прийти к выводу, что программа работает верно. При оценке порядка сходимости на равномерной сетке были получены значения даже больше, чем 1. При оценке порядка сходимости на неравномерной сетке первое значение также больше 1, но вот отношение вложенной сетки к дважды вложенной уже меньше 1. Вероятно, это связано с искажениями сетки (как от изопараметрических преобразований, так и от неравномерности кубической сетки).

5. Код программы

Генерация массивов граней и узлов для краевых условий второго/третьего и первого рода соответственно:

```
void Grid::FormBC(const std::array<std::vector<double>, SIZE_NODE>& xyz, const
std::vector<int>& missingNodes, const std::vector<int>& missingElements)
{
 auto&& [x, y, z] = xyz;

 for (int h = 0; h < m_BC.size(); h++)
 {
  std::array<std::pair<int, int>, SIZE_NODE> limitsEdge =
CalculationLimitsEdge(m_BC[h].boundaries);

  switch (m_BC[h].typeBC)
  {
   case TYPE_BOUNDARY_CONDITION::FIRST:

    for (int k = limitsEdge[2].first; k <= limitsEdge[2].second; k++)
     for (int j = limitsEdge[1].first; j <= limitsEdge[1].second; j++)
      for (int i = limitsEdge[0].first; i <= limitsEdge[0].second; i++)
      {
       int number = k * x.size() * y.size() + j * x.size() + i;
       //коррекция номеров вершин с учётом пропущенных вершин
       number -= missingNodes[number];
       m_BC_1.push_back(number);
      }
    //удаление дубликатов из массива первых краевых(дубликаты расположены на
рёбрах, где грани соприкасаются)
    removeDuplicates(m_BC_1);
    break;

   case TYPE_BOUNDARY_CONDITION::SECOND:
   case TYPE_BOUNDARY_CONDITION::THIRD:
    std::vector<Edge>& BC = (m_BC[h].typeBC == TYPE_BOUNDARY_CONDITION::SECOND) ?
m_BC_2 : m_BC_3;

    bool reducelimit = false;
    int coordinateMatching = 0;
    //коррекция лимитов циклов
    for (int i = 0; i < limitsEdge.size(); i++)
     if (limitsEdge[i].first != limitsEdge[i].second)
      limitsEdge[i].second--;
     else
     {
      coordinateMatching = i;
      if (0 < limitsEdge[i].first)
      {
       std::array<double, SIZE_NODE> leftNode{0, 0, 0};
       leftNode[i]--;
       for (int j = 0; j < leftNode.size(); j++)
        leftNode[j] = (xyz[j][limitsEdge[j].first + leftNode[j]] +
xyz[j][limitsEdge[j].second + leftNode[j]]) / 2.0;
       if (InDomain(leftNode).first)
        reducelimit = true;
      }
     }

    for (int k = limitsEdge[2].first; k <= limitsEdge[2].second; k++)
    {
     int kxy_0 = k * x.size() * y.size();
     int kxy_1 = (k + 1) * x.size() * y.size();

     for (int j = limitsEdge[1].first; j <= limitsEdge[1].second; j++)
     {
      int jx_0 = j * x.size();
      int jx_1 = (j + 1) * x.size();

      for (int i = limitsEdge[0].first; i <= limitsEdge[0].second; i++)
```

```
        {
          std::array<int, NUMBER_NODES_CUBE / 2> vRect;
          int elementOffset = k * (x.size() - 1) * (y.size() - 1) + j * (x.size() -
1) + i;
          //вычисление вершин прямоугольника, который будет разбит на
грани(треугольники)
          switch (coordinateMatching)
          {
          case 0:
            vRect = { kxy_0 + jx_0 + i, kxy_0 + jx_1 + i, kxy_1 + jx_0 + i, kxy_1 +
jx_1 + i };
            if(reducelimit) elementOffset--;
            break;
          case 1:
            vRect = { kxy_0 + jx_0 + i, kxy_0 + jx_0 + i + 1, kxy_1 + jx_0 + i, kxy_1
+ jx_0 + i + 1 };
            if (reducelimit) elementOffset -= x.size() - 1;
            break;
          case 2:
            vRect = { kxy_0 + jx_0 + i, kxy_0 + jx_0 + i + 1, kxy_0 + jx_1 + i, kxy_0
+ jx_1 + i + 1 };
            if (reducelimit) elementOffset -= (x.size() - 1) * (y.size() - 1);
            break;
          }

          //коррекция номеров вершин с учётом пропущенных вершин
          for (int l = 0; l < vRect.size(); l++)
            vRect[l] -= missingNodes[vRect[l]];

          //коррекция номера элемента-куба с учётом пропущенных элементов-кубов
          elementOffset -= missingElements[elementOffset];
          //каждый куб был разбит на тетраэдры
          elementOffset *= m_gridPattern;

          std::array<int, SIZE_EDGE> vertexes;
          if (m_gridPattern == GRID_PATTERN::FIVE)
          {
            if ((i + j + k) % 2 == 0)
            {
              vertexes = { vRect[0], vRect[1], vRect[3] };
              BC.push_back({ vertexes, SearchElement(vertexes, elementOffset,
elementOffset + m_gridPattern - 1) });
              vertexes = { vRect[0], vRect[2], vRect[3] };
              BC.push_back({ vertexes, SearchElement(vertexes, elementOffset,
elementOffset + m_gridPattern - 1) });
            }
            else
            {
              vertexes = { vRect[0], vRect[1], vRect[2] };
              BC.push_back({ vertexes, SearchElement(vertexes, elementOffset,
elementOffset + m_gridPattern - 1) });
              vertexes = { vRect[1], vRect[2], vRect[3] };
              BC.push_back({ vertexes, SearchElement(vertexes, elementOffset,
elementOffset + m_gridPattern - 1) });
            }
          }
          else
          {
            vertexes = { vRect[0], vRect[1], vRect[2] };
            BC.push_back({ vertexes, SearchElement(vertexes, elementOffset,
elementOffset + m_gridPattern - 1) });
            vertexes = { vRect[1], vRect[2], vRect[3] };
            BC.push_back({ vertexes, SearchElement(vertexes, elementOffset,
elementOffset + m_gridPattern - 1) });
          }
```

```
        }
       }
      }

      break;
   }
  }

}
```

Построение портрета глобальной матрицы:

```
void SparseSLAE::GenerateSLAE(const std::vector<Grid::Element>& elements, int N, int
neighboringNodes)
{
    int memory = N * neighboringNodes;
    std::vector<std::vector<int>> list(2, std::vector<int>(memory, 0));

    std::vector<int>& ig = m_M.ig;
    ig.reserve(N);
    std::vector<int>& jg = m_M.jg;
    jg.reserve(memory);
    std::vector<int> listbeg(N, 0);
    int listSize = 0;

    for (const auto& elem : elements)
    {
        for (int i = 0; i < elem.vertexes.size(); i++)
        {
            int k = elem.vertexes[i];

            for (int j = i + 1; j < elem.vertexes.size(); j++)
            {
                int ind1 = k;

                int ind2 = elem.vertexes[j];
                if (ind2 < ind1)
                {
                    ind1 = ind2;
                    ind2 = k;
                }
                int iaddr = listbeg[ind2];
                if (!iaddr)
                {
                    listSize++;
                    listbeg[ind2] = listSize;
                    list[0][listSize] = ind1;
                    list[1][listSize] = 0;
                }
                else
                {
                    while (list[0][iaddr] < ind1 && list[1][iaddr] > 0)
                    {
                        iaddr = list[1][iaddr];
                    }
                    if (list[0][iaddr] > ind1)
                    {
                        listSize++;
                        list[0][listSize] = list[0][iaddr];
                        list[1][listSize] = list[1][iaddr];
                        list[0][iaddr] = ind1;
                        list[1][iaddr] = listSize;
                    }
                    else
```

```
                    {
                        if (list[0][iaddr] < ind1)
                        {
                            listSize++;
                            list[1][iaddr] = listSize;
                            list[0][listSize] = ind1;
                            list[1][listSize] = 0;
                        }
                    }
                }
            }
        }
    }

    ig.push_back(0);
    for (int i = 0; i < N; i++)
    {
        ig.push_back(ig[i]);
        int iaddr = listbeg[i];
        while (iaddr != 0)
        {
            jg.push_back(list[0][iaddr]);
            ig[i + 1]++;
            iaddr = list[1][iaddr];
        }
    }

    m_M.ggl.resize(m_M.jg.size());
    m_M.ggu.resize(m_M.jg.size());
    m_M.di.resize(N);
    m_b.resize(N);
    m_x.resize(N);
}
```

Занесение локальных матриц и векторов в глобальную матрицу и вектор:

```
template<int n>
void FEM::AddFiniteVector(const FiniteVector<n>& lV, const std::array<int, n>& L)
{
 Vector& b = m_sparseSLAE.Set_b();
 for (int i = 0; i < n; i++)
  b[L[i]] += lV[i];
}

template<int n>
void FEM::AddFiniteMatrix(const FiniteMatrix<n>& lM, const std::array<int, n>& L)
{
 SparseSLAE::SparseMatrix& M = m_sparseSLAE.Set_M();

 for (int i = 0; i < n; i++)
 {
  M.di[L[i]] += lM[i][i];

  int ibeg = M.ig[L[i]];
  int iend = M.ig[L[i] + 1];
  for (int j = 0; j < i; j++)
  {
   int index = binarySearch(M.jg, L[j], ibeg, iend - 1);

   M.ggu[index] += lM[j][i];
   M.ggl[index] += lM[i][j];
   ibeg++;
  }
```

```
  }
}
```