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 № 4

по дисциплине «КТМиАД»

МЕТОДЫ ПОСТРОЕНИЯ ИЗОБРАЖЕНИЙ КОНЕЧНОЭЛЕМЕНТНЫХ РЕШЕНИЙ

Бригада 10 БАРАНОВ ЯРОСЛАВ

Группа ПММ-32 МАКАРЫЧЕВ СЕРГЕЙ

Преподаватели КОШКИНА Ю.И.

Новосибирск, 2023

1. Задание

Цель: изучить методы построения изображений конечноэлементного решения при решении различных задач.

Вариант: построение сечений для трёхмерного скалярного поля с помощью изолиний и заливки цветом.

2. Особенности реализации и возможности

Программа реализована с помощью библиотеки языка Python - matplotlib.

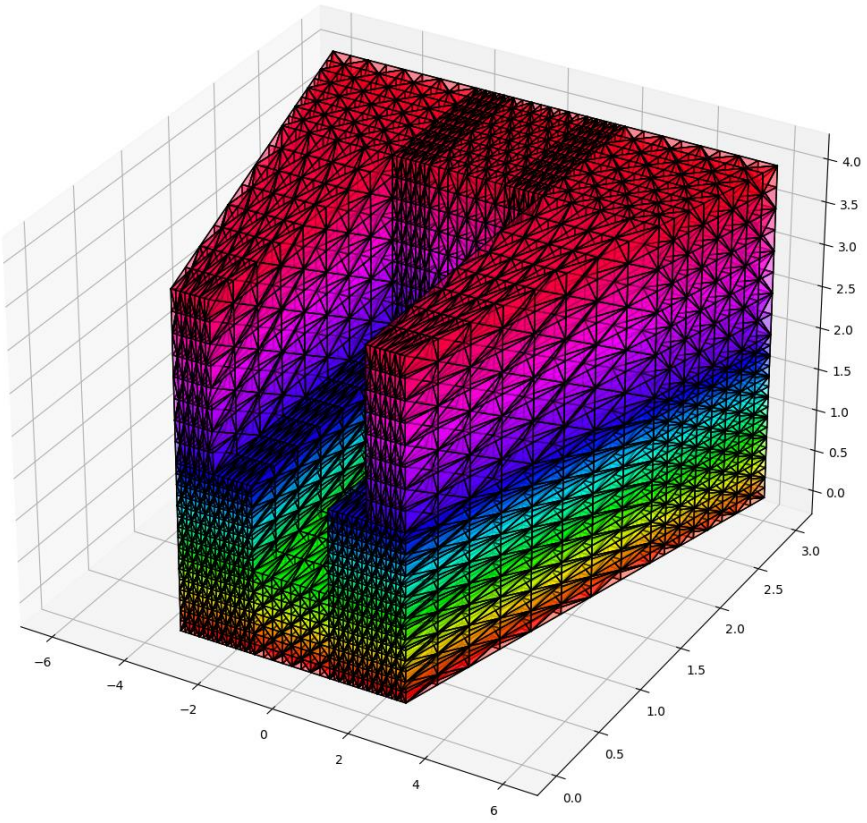
Учтена геометрия расчётной области путём маскировки треугольников, получаемых в результате работы функции построения изолиний tricontourf из matplotlib (проверяется попадание центра треугольника в шестигранные подобласти путём определения по какую сторону от граней он лежит).

Поддерживаются отрисовки сечений вдоль трёх основных плоскостей: Oyz, Oxz, Oxy. Имеется возможность динамически менять значения x, y, z с помощью соответствующих слайдеров. Благодаря чему возможно просматривать слои решения по соответствующей координате не перезапуская программу.

В общем случае сечением может быть ребро или несколько ребер, эти случаи не поддерживаются.

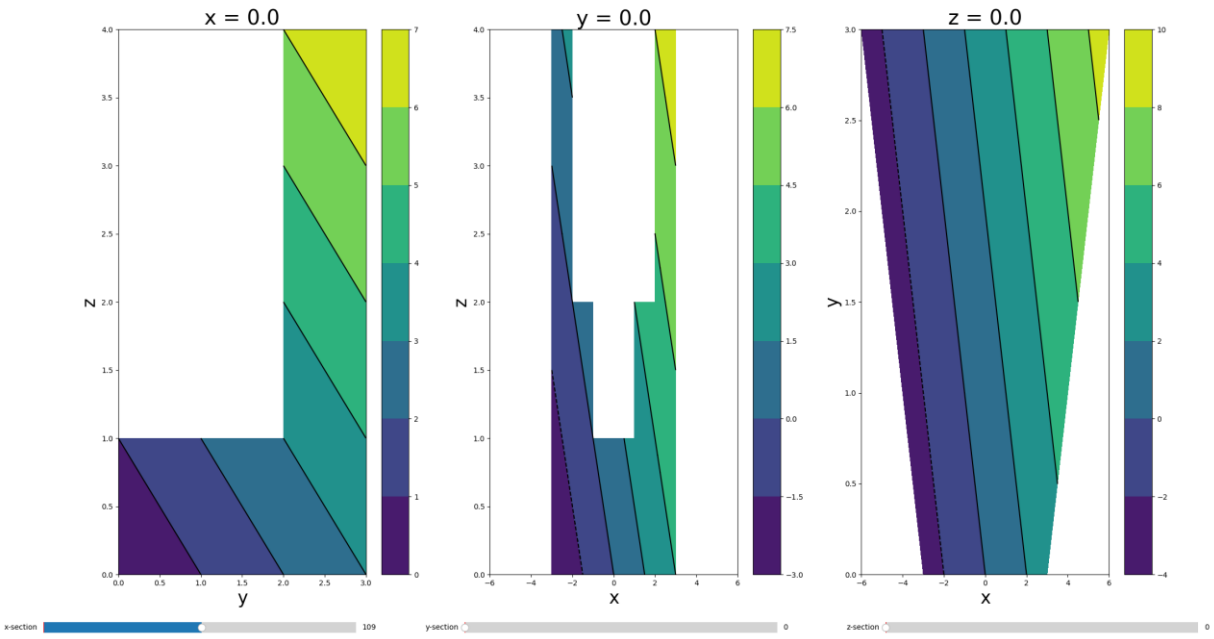
3. Тестирование

Расчётная область:

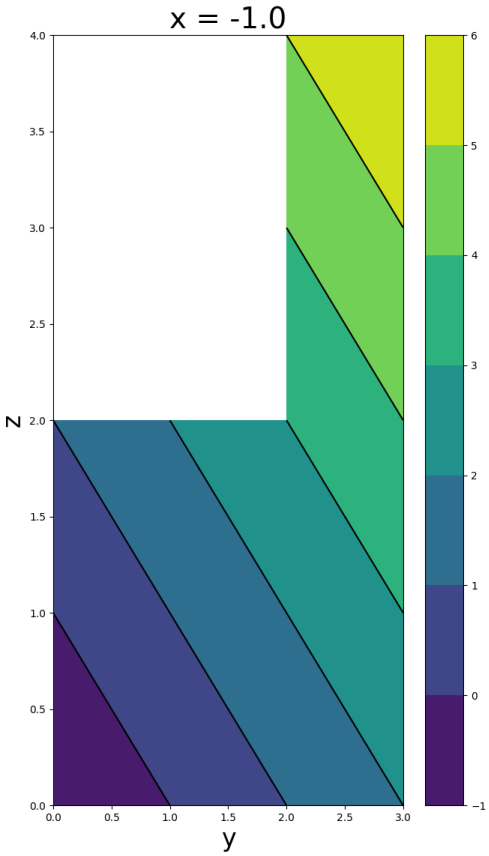


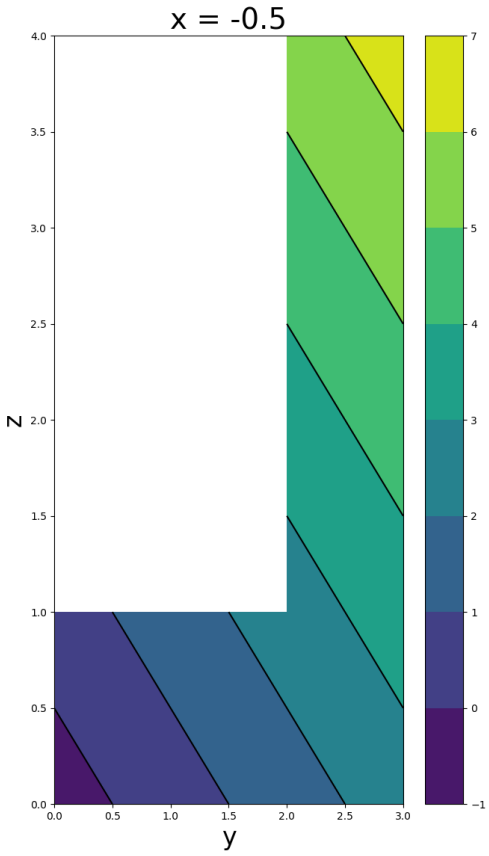
Пусть приближаемая функция: $u = x + y + z$.

Общий вид программы отрисовки сечений:

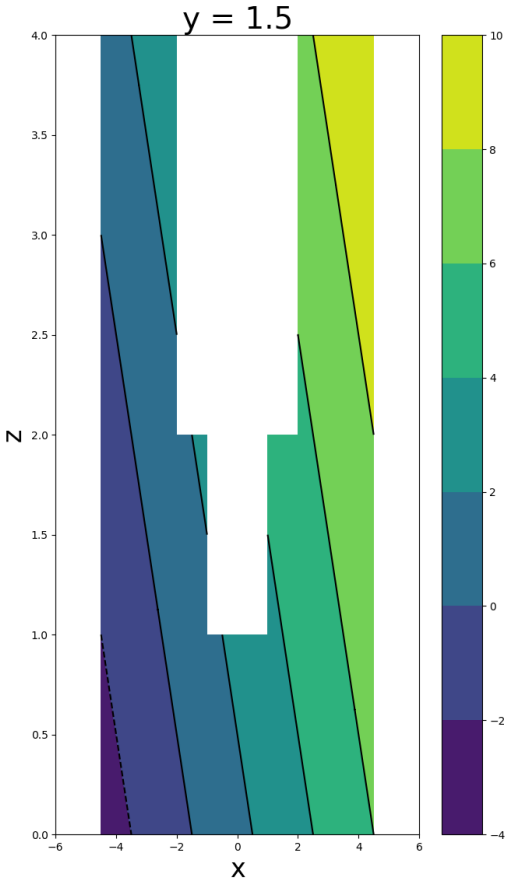


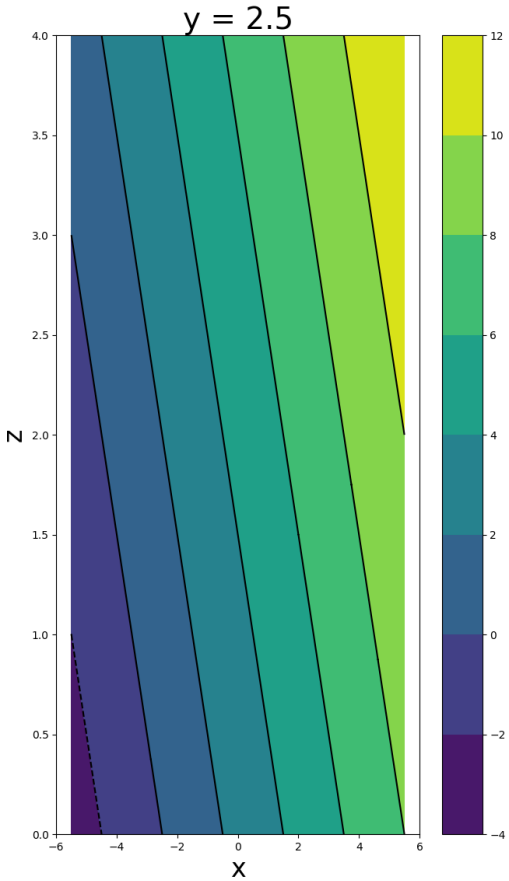
Некоторые слои по x:



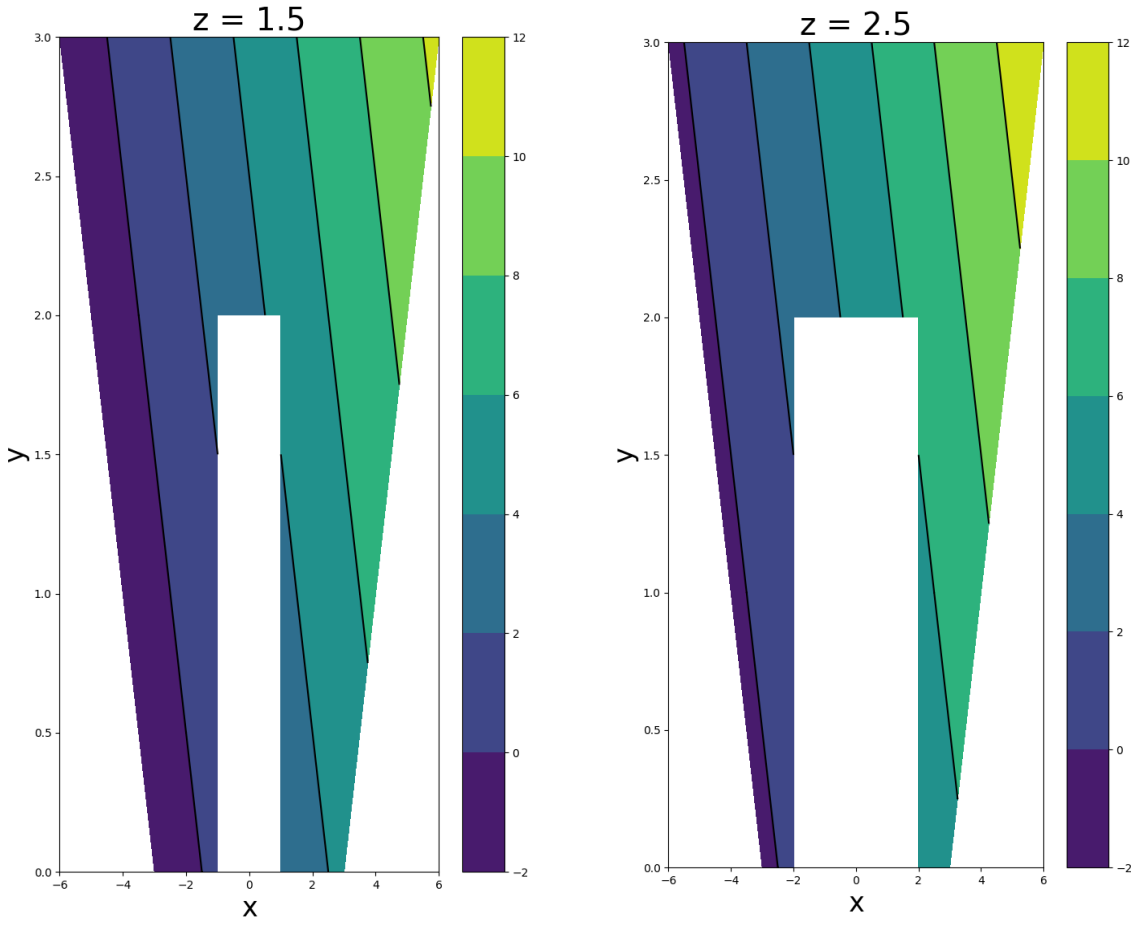


Некоторые слои по y:



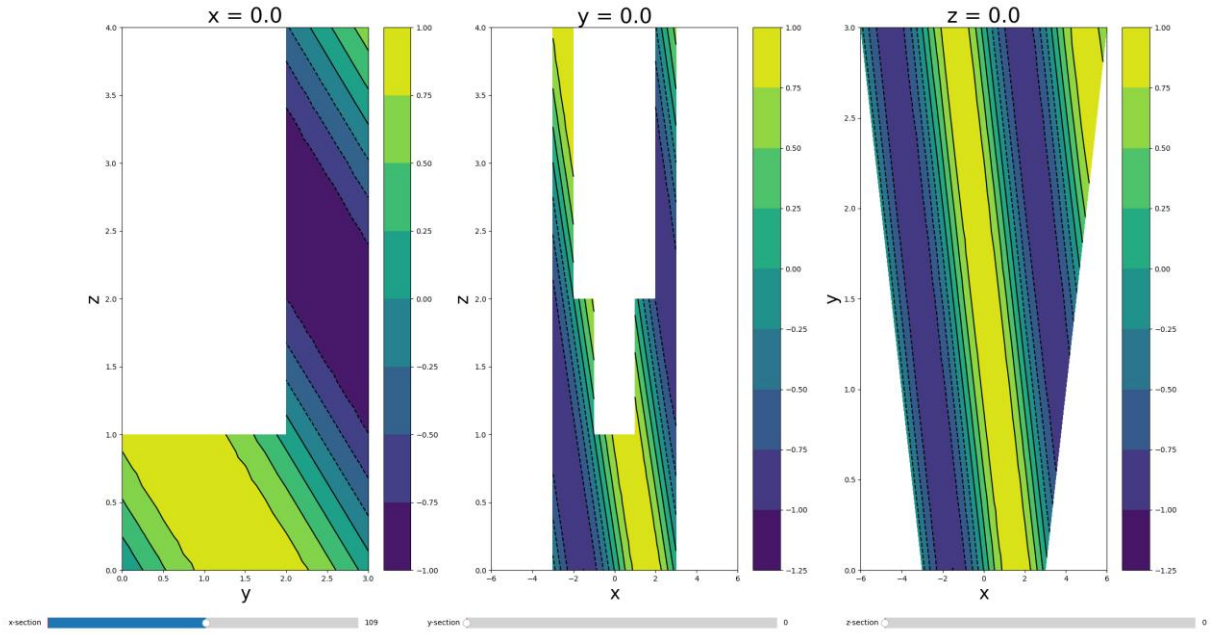


Некоторые слои по z:

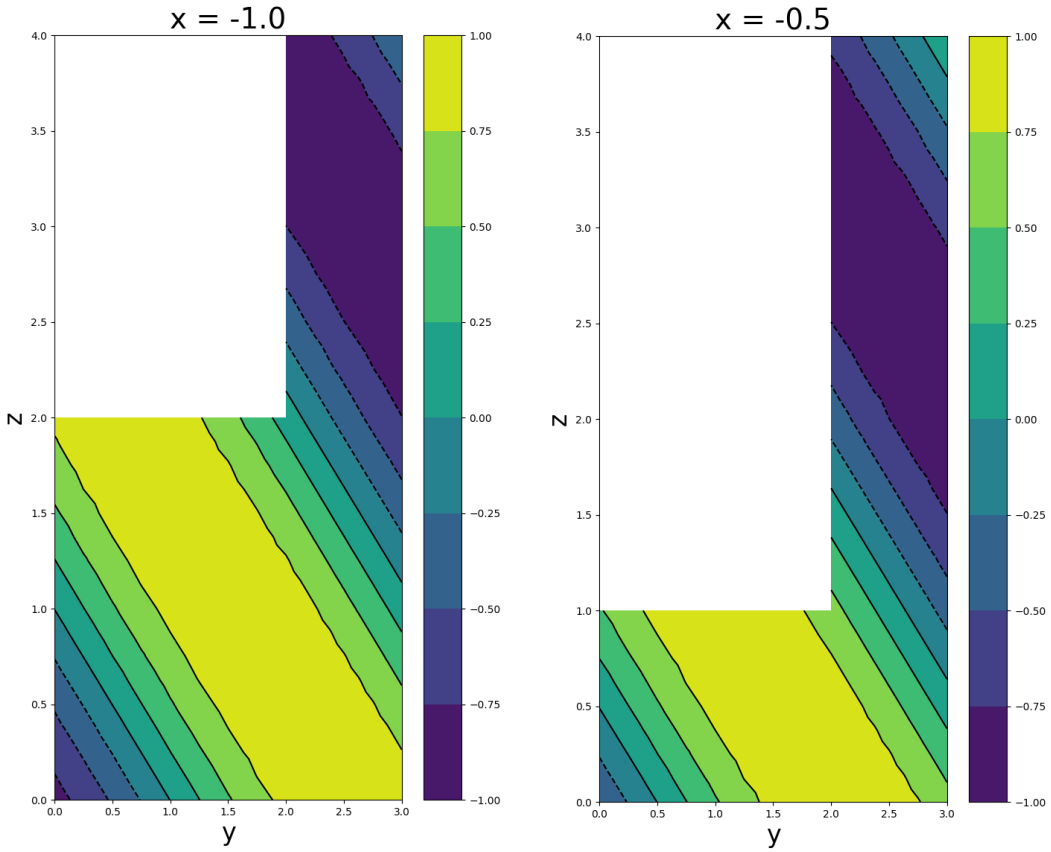


Пусть приближаемая функция: $u = \sin(x + y + z)$.

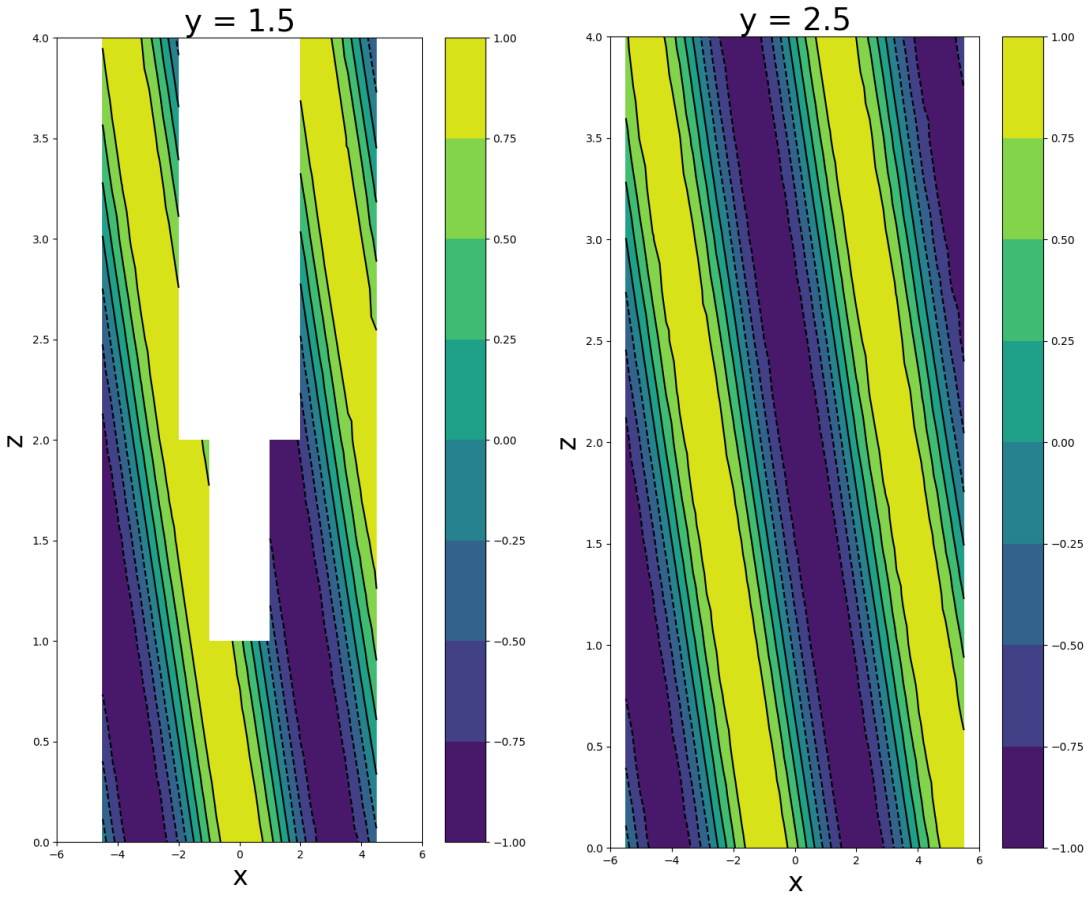
Общий вид программы отрисовки сечений:



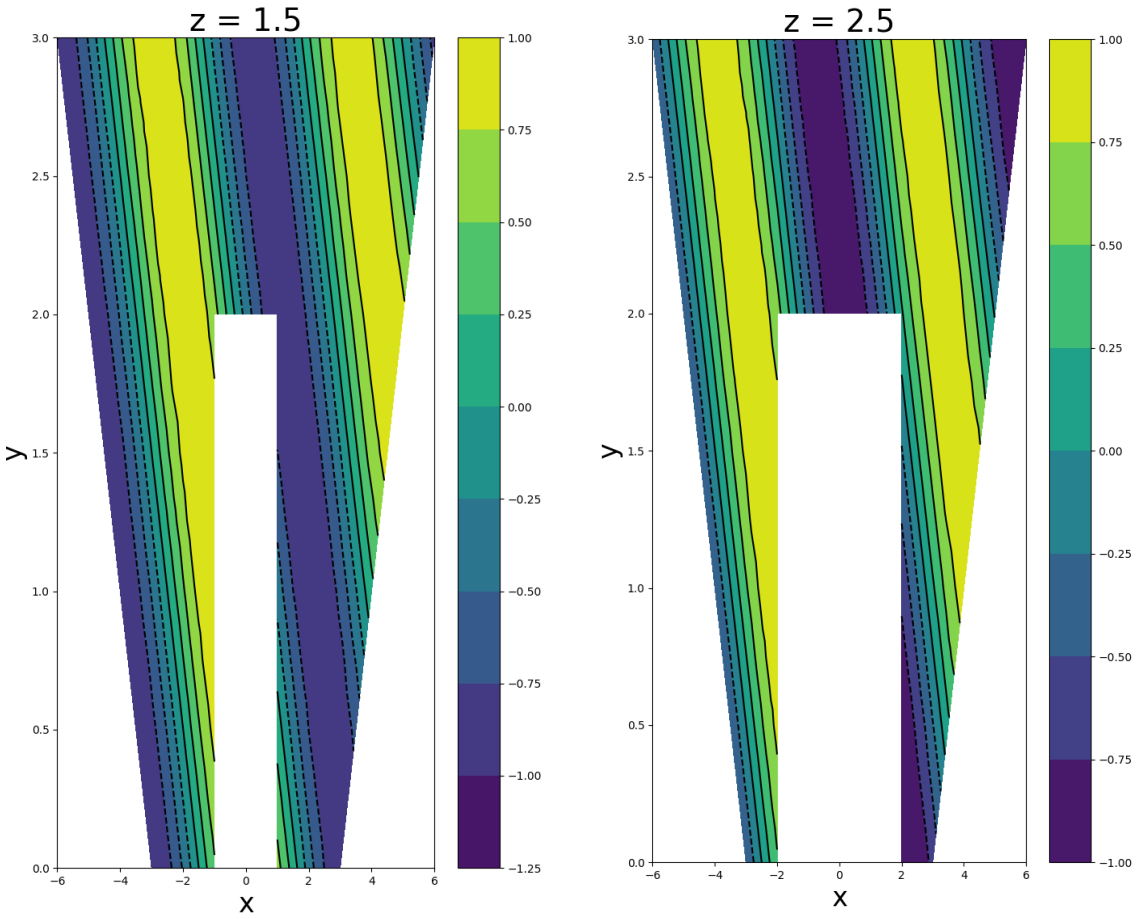
Некоторые слои по x:



Некоторые слои по y:



Некоторые слои по z:



4. Выводы

Зрительно сравнивая сечения расчетной области с полученными, можно сделать вывод, что они рисуются верно. На некоторых рисунках видно, что область определения сечения не является выпуклой.

5. Код программы

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
from matplotlib.widgets import Button, Slider
import matplotlib.tri as mtri


def SameSide(v1, v2, v3, v4, p):
    normal = np.cross(v2-v1, v3-v1)
    return (np.dot(normal, v4-v1) * np.dot(normal, p-v1) >= 0)


def pointInside(verts, p):
    return SameSide(verts[1], verts[0], verts[3], verts[5], p) and
SameSide(verts[5], verts[4], verts[7], verts[1], p) and SameSide(verts[0], verts[1],
verts[4], verts[2], p) and SameSide(verts[3], verts[2], verts[7], verts[1], p) and
```

```
SameSide(verts[1], verts[3], verts[5], verts[0], p) and SameSide(verts[0], verts[4],
verts[2], verts[1], p)


x = []
y = []
z = []
values = []
subdomains = []

with open('../../res/output/solution.txt', 'r') as f:
    for line in f.readlines():
        (xt, yt, zt, valuest) = line.split()
        x.append(float(xt))
        y.append(float(yt))
        z.append(float(zt))
        values.append(float(valuest))

with open('../../res/output/hexahedron subdomains.txt', 'r') as f:
    for line in f.readlines():
        subdomain = line.split()
        nodes = []
        for i in range(8):
            node = []
            for j in range(3):
                node.append(float(subdomain[3 * i + j]))
            nodes.append(node)
        subdomains.append(nodes)

unique_x = sorted(list(set(x)))
unique_y = sorted(list(set(y)))
unique_z = sorted(list(set(z)))

fig = plt.figure()

#-------------------------------------------------------------------------------
---------------

def getMaskx(ymid, zmid, unx):
    mask = []
    for i in range(len(ymid)):
        flag = True
        p = [unx, ymid[i], zmid[i]]
        for j in range(len(subdomains)):
            if pointInside(np.array(subdomains[j]), np.array(p)):
                flag = False
                break
        mask.append(flag)
    return mask

def getDatax(xVal):
    y_graph = []
    z_graph = []
    values_graph = []

    for i in range(len(x)):
        if(x[i]==xVal):
            z_graph.append(z[i])
            y_graph.append(y[i])
            values_graph.append(values[i])

    return y_graph, z_graph, values_graph


axx = fig.add_subplot(1, 3, 1)
```

```
axx.set_xlim([unique_y[0], unique_y[len(unique_y)-1]])
axx.set_ylim([unique_z[0], unique_z[len(unique_z)-1]])
axx.set_xlabel('y', fontsize=24)
axx.set_ylabel('z', fontsize=24)

axx_slider = fig.add_axes([0.07, 0.02, 0.23, 0.03])
x_slider = Slider(
    ax=axx_slider,
    label='x-section',
    valmin=0,
    valmax=len(unique_x) - 1,
    valinit=0,
    valfmt="%i"
)


for i in range(len(unique_x)):
    try:
        datax = getDatax(unique_x[i])

        triangx = mtri.Triangulation(datax[0] , datax[1])

        ymidx = np.array(datax[0])[triangx.triangles].mean(axis=1)
        zmidx = np.array(datax[1])[triangx.triangles].mean(axis=1)
        triangx.set_mask(getMaskx(ymidx, zmidx, unique_x[i]))

        countourx = axx.tricontourf(triangx, datax[2])
        axx.tricontour(countourx, colors='black')
        colbarx = fig.colorbar(countourx, ax=axx)
        axx.set_title('x = ' + str(unique_x[i]), fontsize=28)
        break
    except:
        pass


def updatex(val):
    try:
        data = getDatax(unique_x[int(val)])

        triang = mtri.Triangulation(data[0] , data[1])

        ymid = np.array(data[0])[triang.triangles].mean(axis=1)
        zmid = np.array(data[1])[triang.triangles].mean(axis=1)
        triang.set_mask(getMaskx(ymid, zmid, unique_x[int(val)]))

        axx.collections.clear()
        countour = axx.tricontourf(triang, data[2])
        axx.tricontour(countour, colors='black')
        global colbarx
        colbarx.remove()
        colbarx = fig.colorbar(countour, ax=axx)
        axx.set_title('x = ' + str(unique_x[int(val)]), fontsize=28)
        plt.draw()
    except:
        pass


x_slider.on_changed(updatex)

#-------------------------------------------------------------------------------
---------------
def getMasky(xmid, zmid, uny):
    mask = []
    for i in range(len(xmid)):
        flag = True
```

```
            p = [xmid[i], uny, zmid[i]]
            for j in range(len(subdomains)):
                if pointInside(np.array(subdomains[j]), np.array(p)):
                    flag = False
                    break
            mask.append(flag)
        return mask

    def getDatay(yVal):
        x_graph = []
        z_graph = []
        values_graph = []

        for i in range(len(y)):
            if(y[i]==yVal):
                x_graph.append(x[i])
                z_graph.append(z[i])
                values_graph.append(values[i])

        return x_graph, z_graph, values_graph

    axy = fig.add_subplot(1, 3, 2)
    axy.set_xlim([unique_x[0], unique_x[len(unique_x)-1]])
    axy.set_ylim([unique_z[0], unique_z[len(unique_z)-1]])
    axy.set_xlabel('x', fontsize=24)
    axy.set_ylabel('z', fontsize=24)

    axy_slider = fig.add_axes([0.38, 0.02, 0.23, 0.03])
    y_slider = Slider(
        ax=axy_slider,
        label='y-section',
        valmin=0,
        valmax=len(unique_y) - 1,
        valinit=0,
        valfmt="%i"
    )
    for i in range(len(unique_y)):
        try:
            datay = getDatay(unique_y[i])

            triangy = mtri.Triangulation(datay[0] , datay[1])

            xmidy = np.array(datay[0])[triangy.triangles].mean(axis=1)
            ymidy = np.array(datay[1])[triangy.triangles].mean(axis=1)
            triangy.set_mask(getMasky(xmidy, ymidy, unique_y[i]))

            countoury = axy.tricontourf(triangy, datay[2])
            axy.tricontour(countoury, colors='black')
            colbary = fig.colorbar(countoury, ax=axy)
            axy.set_title('y = ' + str(unique_y[i]), fontsize=28)
            break
        except:
            pass

    def updatey(val):
        try:
            data = getDatay(unique_y[int(val)])

            triang = mtri.Triangulation(data[0] , data[1])

            xmid = np.array(data[0])[triang.triangles].mean(axis=1)
            ymid = np.array(data[1])[triang.triangles].mean(axis=1)
            triang.set_mask(getMasky(xmid, ymid, unique_y[int(val)]))

            axy.collections.clear()
```

```
        countour = axy.tricontourf(triang, data[2])
        axy.tricontour(countour, colors='black')
        global colbary
        colbary.remove()
        colbary = fig.colorbar(countour, ax=axy)
        axy.set_title('y = ' + str(unique_y[int(val)]), fontsize=28)
        plt.draw()
    except:
        pass

y_slider.on_changed(updatey)

#--------------------------------------------------------------------------------
---------------
def getMaskz(xmid, ymid, unz):
    mask = []
    for i in range(len(xmid)):
        flag = True
        p = [xmid[i], ymid[i], unz]
        for j in range(len(subdomains)):
            if pointInside(np.array(subdomains[j]), np.array(p)):
                flag = False
                break
        mask.append(flag)
    return mask

def getDataz(zVal):
    x_graph = []
    y_graph = []
    values_graph = []

    for i in range(len(z)):
        if(z[i]==zVal):
            x_graph.append(x[i])
            y_graph.append(y[i])
            values_graph.append(values[i])

    return x_graph, y_graph, values_graph

axz = fig.add_subplot(1, 3, 3)
axz.set_xlim([unique_x[0], unique_x[len(unique_x)-1]])
axz.set_ylim([unique_y[0], unique_y[len(unique_y)-1]])
axz.set_xlabel('x', fontsize=24)
axz.set_ylabel('y', fontsize=24)

axz_slider = fig.add_axes([0.69, 0.02, 0.23, 0.03])
z_slider = Slider(
    ax=axz_slider,
    label='z-section',
    valmin=0,
    valmax=len(unique_z) - 1,
    valinit=0,
    valfmt="%i"
)

for i in range(len(unique_z)):
    try:
        dataz = getDataz(unique_z[i])

        triangz = mtri.Triangulation(dataz[0] , dataz[1])

        xmidz = np.array(dataz[0])[triangz.triangles].mean(axis=1)
        ymidz = np.array(dataz[1])[triangz.triangles].mean(axis=1)
        triangz.set_mask(getMaskz(xmidz, ymidz, unique_z[i]))
```

```
            countourz = axz.tricontourf(triangz, dataz[2])
            axz.tricontour(countourz, colors='black')
            colbarz = fig.colorbar(countourz, ax=axz)
            axz.set_title('z = ' + str(unique_z[i]), fontsize=28)
            break
        except:
            pass

    def updatez(val):
        try:
            data = getDataz(unique_z[int(val)])

            triang = mtri.Triangulation(data[0] , data[1])

            xmid = np.array(data[0])[triang.triangles].mean(axis=1)
            ymid = np.array(data[1])[triang.triangles].mean(axis=1)
            triang.set_mask(getMaskz(xmid, ymid, unique_z[int(val)]))

            axz.collections.clear()
            countour = axz.tricontourf(triang, data[2])
            axz.tricontour(countour, colors='black')
            global colbarz
            colbarz.remove()
            colbarz = fig.colorbar(countour, ax=axz)
            axz.set_title('z = ' + str(unique_z[int(val)]), fontsize=28)
            plt.draw()
        except:
            pass

    z_slider.on_changed(updatez)

    plt.show()
```