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. Реализовать методы построения изображения решения для элементов высоких порядков.

*Вариант: построение сечений для трёхмерного скалярного поля.*

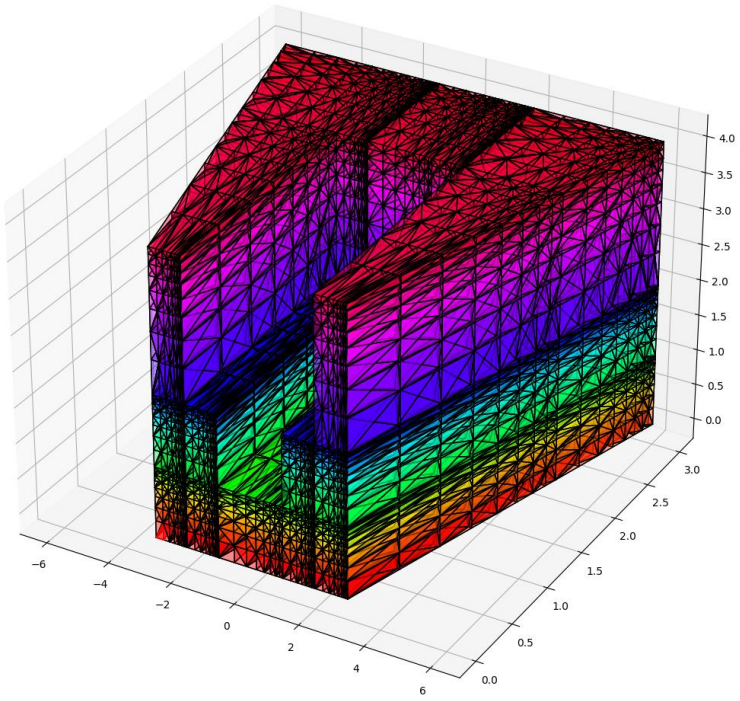
## 2. Особенности реализации и возможности

Программа реализована с помощью библиотеки языка Python - matplotlib. Формат графика: точечный. Данный формат был выбран в связи с тем, что область определения сечений в общем случае не является выпуклой. Поэтому нужен был простой способ отображения решения без построения двумерной триангуляции для отрисовки графика.

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.

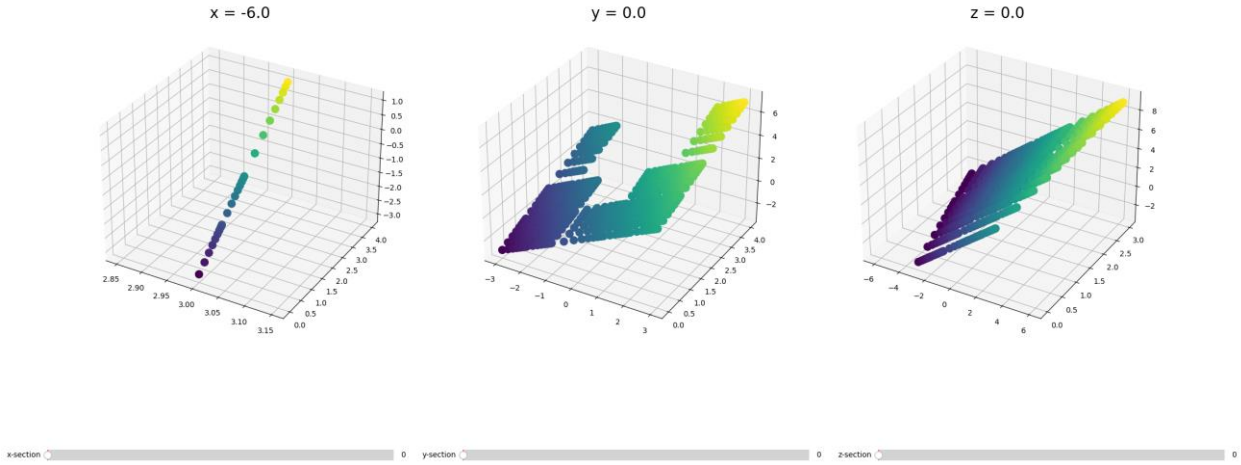
## 3. Тестирование

Расчётная область:

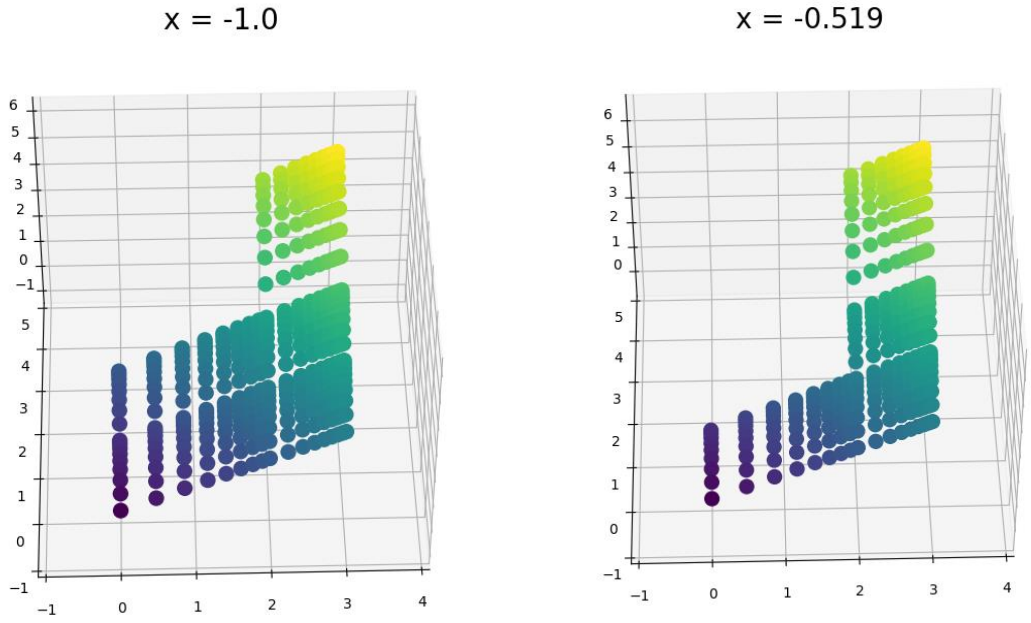


Пусть приближаемая функция: $u = x + y + z$.

Начальное состояние отрисовки сечений при первых слоях по x, y, z:

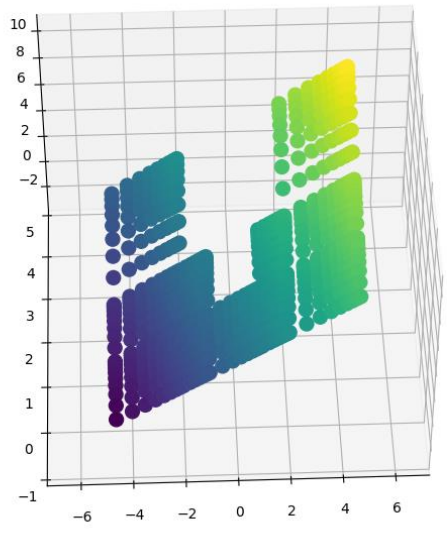



Некоторые слои по x:

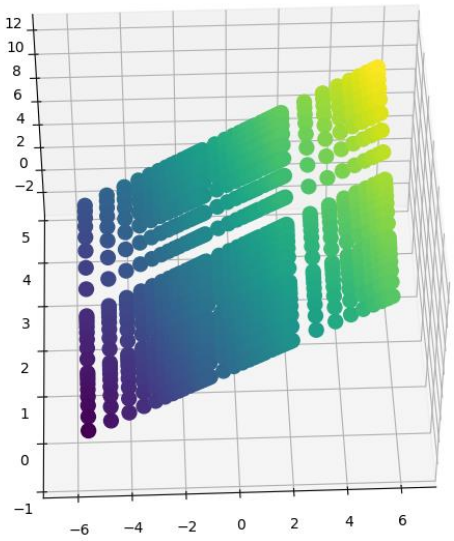



Некоторые слои по y:

y = 1.616

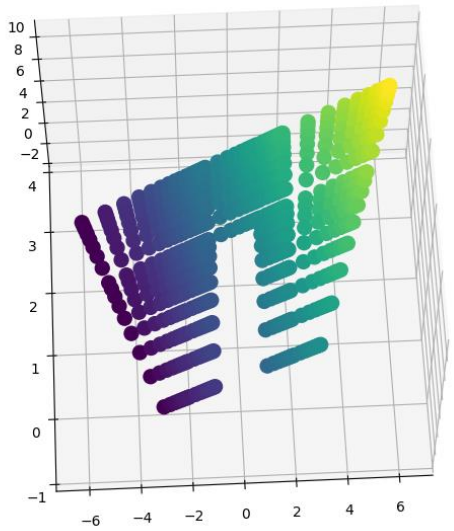


y = 2.586

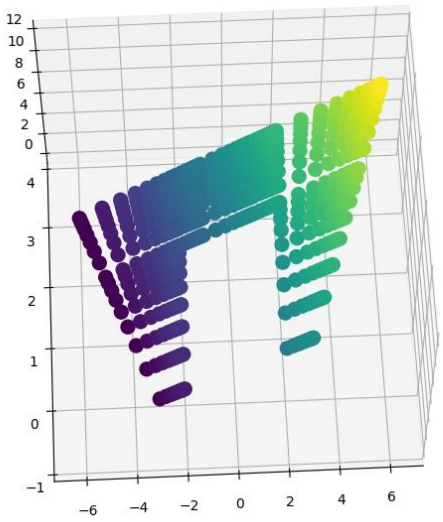


Некоторые слои по z:

z = 1.586

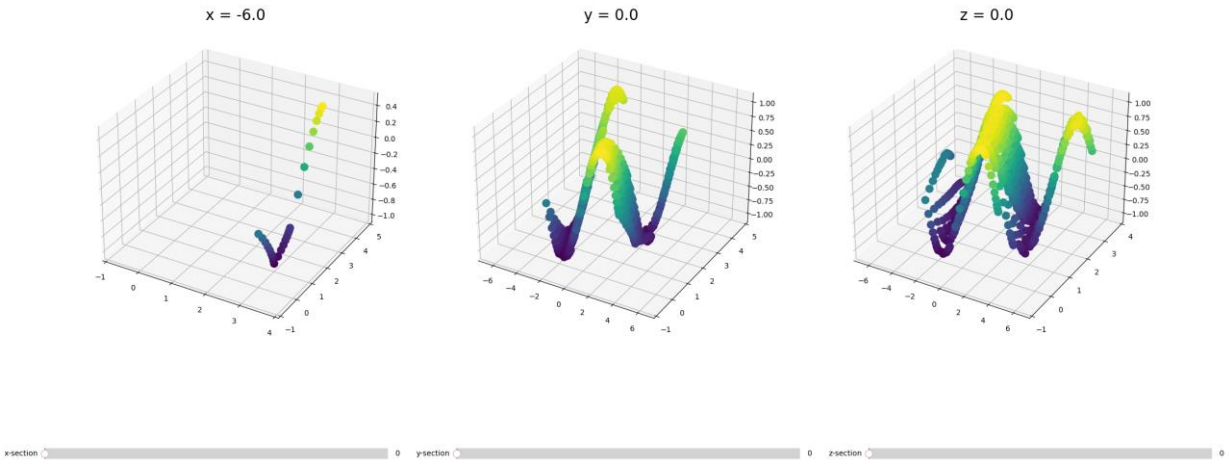


z = 2.865

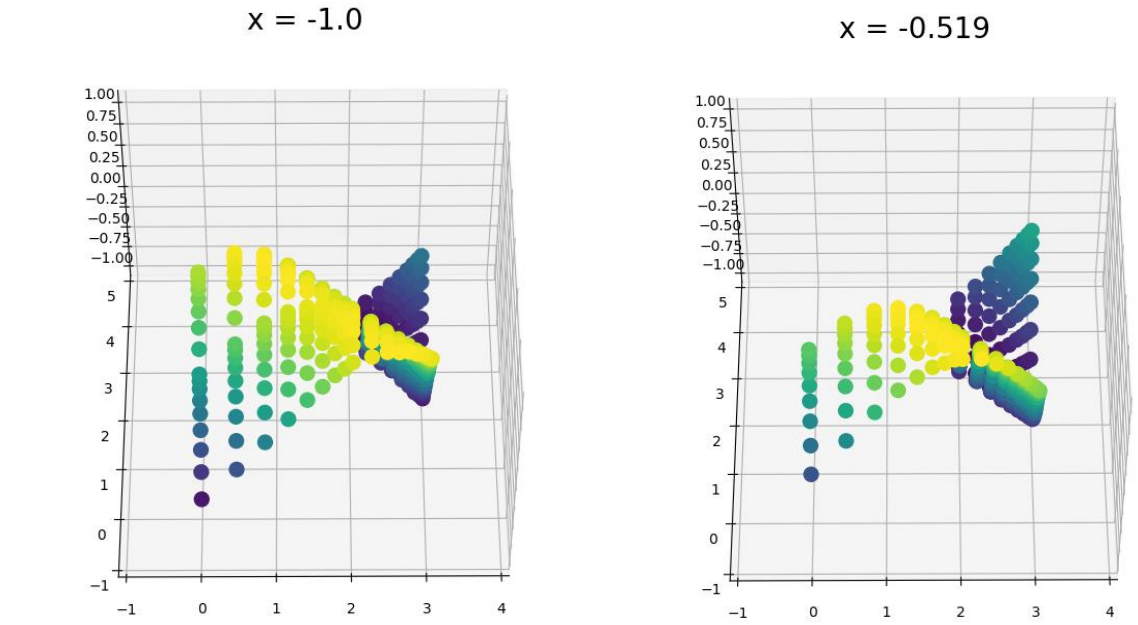


Пусть приближаемая функция: $u = \sin(x + y + z)$.

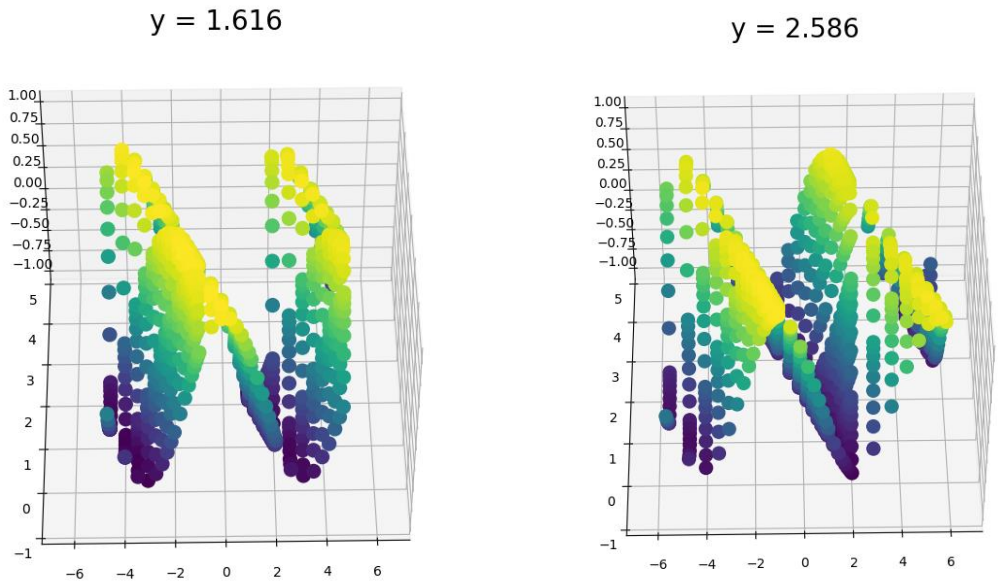
Начальное состояние отрисовки сечений при первых слоях по x, y, z:




Некоторые слои по x:

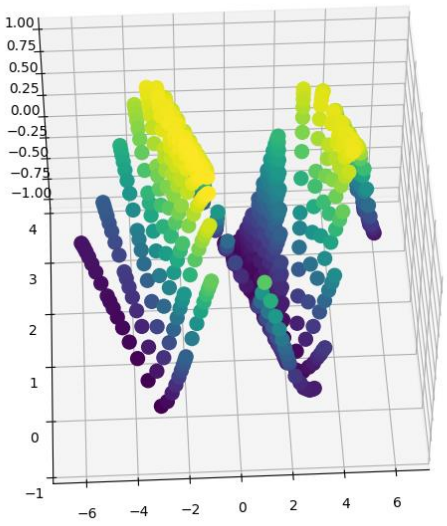



Некоторые слои по y:

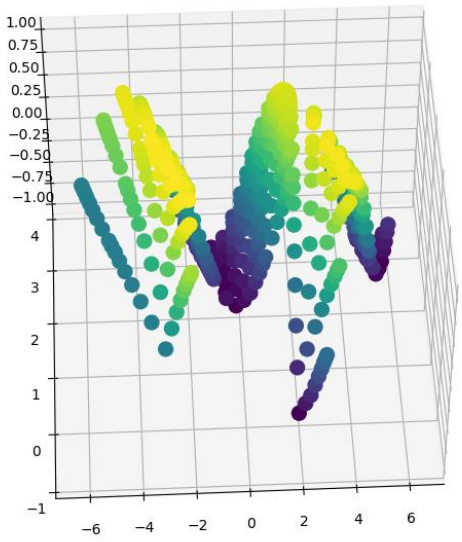


Некоторые слои по z:

z = 1.586




z = 2.865




## 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, поэтому часть узлов в отображаемом сечении приближаемой функции отсутствуют.

## 5. Код программы

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

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

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))


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

fig = plt.figure()
```

```
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, projection='3d')

axx.set_xlim([unique_y[0]-1, unique_y[len(unique_y)-1]+1])
axx.set_ylim([unique_z[0]-1, unique_z[len(unique_z)-1]+1])

axx_slider = fig.add_axes([0.1, 0.1, 0.25, 0.03])
x_slider = Slider(
    ax=axx_slider,
    label='x-section',
    valmin=0,
    valmax=len(unique_x) - 1,
    valinit=0,
    valfmt="%i"
)


datax = getDatax(unique_x[0])
surfacex = axx.scatter(datax[0], datax[1], datax[2], c=datax[2], cmap='viridis', s =
100, alpha = 1)
axx.set_title('x = ' + str(unique_x[0]), fontsize=20)

def updatex(val):
    data = getDatax(unique_x[int(val)])
    global surfacex
    surfacex.remove()
    surfacex = axx.scatter(data[0], data[1], data[2], c=data[2], cmap='viridis', s =
100, alpha = 1)
    axx.set_title('x = ' + str(unique_x[int(val)]), fontsize=20)
    plt.draw()

x_slider.on_changed(updatex)

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

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, projection='3d')

axy.set_xlim([unique_x[0]-1, unique_x[len(unique_x)-1]+1])
axy.set_ylim([unique_z[0]-1, unique_z[len(unique_z)-1]+1])

axy_slider = fig.add_axes([0.4, 0.1, 0.25, 0.03])
y_slider = Slider(
    ax=axy_slider,
    label='y-section',
    valmin=0,
    valmax=len(unique_y) - 1,
    valinit=0,
    valfmt="%i"
)


datay = getDatay(unique_y[0])
surfacey = axy.scatter(datay[0], datay[1], datay[2], c=datay[2], cmap='viridis', s =
100, alpha = 1)
axy.set_title('y = ' + str(unique_y[0]), fontsize=20)

def updatey(val):
    data = getDatay(unique_y[int(val)])
    global surfacey
    surfacey.remove()
    surfacey = axy.scatter(data[0], data[1], data[2], c=data[2], cmap='viridis', s =
100, alpha = 1)
    axy.set_title('y = ' + str(unique_y[int(val)]), fontsize=20)
    plt.draw()

y_slider.on_changed(updatey)


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


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, projection='3d')

axz.set_xlim([unique_x[0]-1, unique_x[len(unique_x)-1]+1])
axz.set_ylim([unique_y[0]-1, unique_y[len(unique_y)-1]+1])

axz_slider = fig.add_axes([0.7, 0.1, 0.25, 0.03])
z_slider = Slider(
    ax=axz_slider,
    label='z-section',
    valmin=0,
    valmax=len(unique_z) - 1,
    valinit=0,
    valfmt="%i"
```

```
)


dataz = getDataz(unique_z[0])
surfacez = axz.scatter(dataz[0], dataz[1], dataz[2], c=dataz[2], cmap='viridis', s =
100, alpha = 1)
axz.set_title('z = ' + str(unique_z[0]), fontsize=20)

def updatez(val):
    data = getDataz(unique_z[int(val)])
    global surfacez
    surfacez.remove()
    surfacez = axz.scatter(data[0], data[1], data[2], c=data[2], cmap='viridis', s =
100, alpha = 1)
    axz.set_title('z = ' + str(unique_z[int(val)]), fontsize=20)
    plt.draw()

z_slider.on_changed(updatez)

plt.show()
```