МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 3.6

«Построение 3D графиков. Работа с mplot3d Toolkit»

по дисциплине «Основы программной инженерии»

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » апреля 2023 г.

Подпись студента_____

Работа защищена

« »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2023

Цель работы:

Исследовать базовые возможности визуализации данных в трехмерном пространстве средствами библиотеки matplotlib языка программирования Python.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: https://github.com/afk552/trolab_6

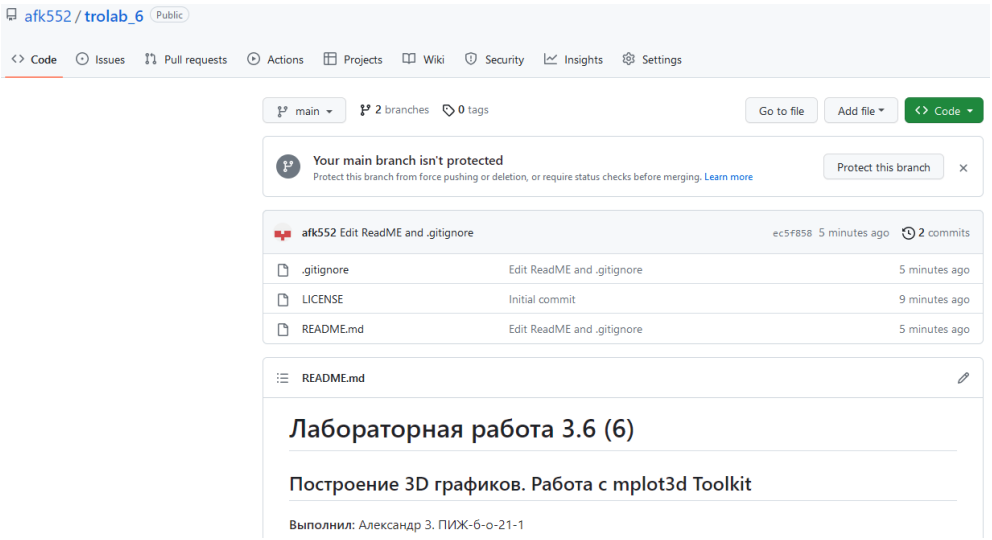


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

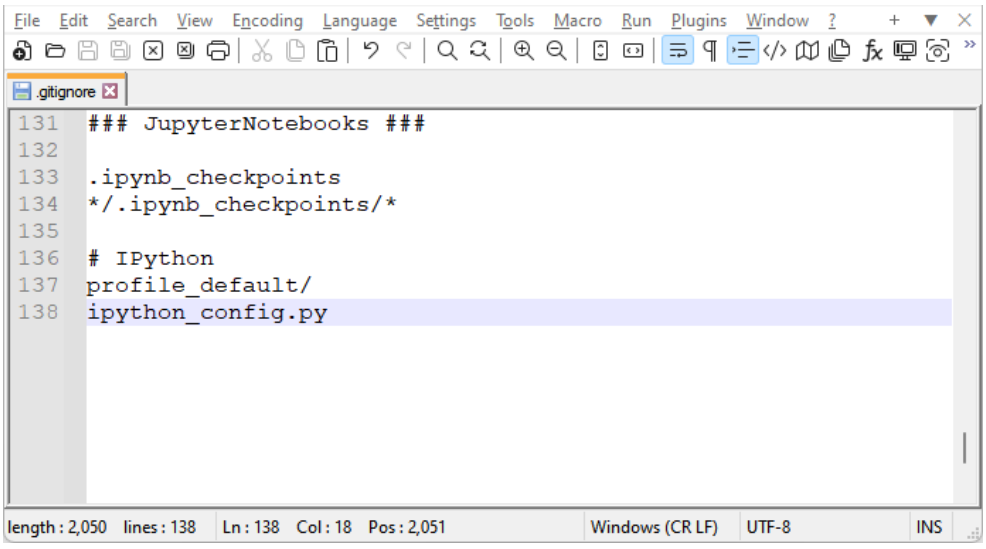


Рисунок 2 – Окно блокнота

Организуйте свой репозиторий в соответствие с моделью ветвления git-flow, рисунок 3.

```
* develop
  main
```

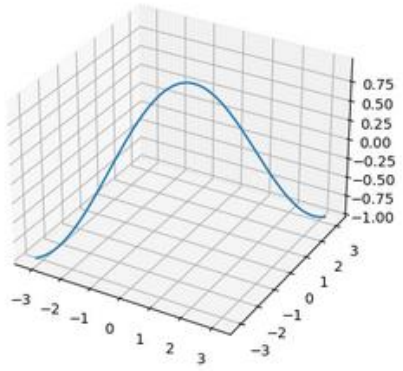
Рисунок 3 – Окно командной строки

Проработать примеры лабораторной работы в отдельном ноутбуке.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        %matplotlib inline
```

Линейный график

```
In [2]: x = np.linspace(-np.pi, np.pi, 50)
        y = x
        z = np.cos(x)
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.plot(x, y, z, label='parametric curve')
Out[2]: [<mpl_toolkits.mplot3d.art3d.Line3D at 0x210ee9d5af0>]
```



Точечный график

```
In [3]: np.random.seed(123)
        x = np.random.randint(-5, 5, 40)
        y = np.random.randint(0, 10, 40)
        z = np.random.randint(-5, 5, 40)
        s = np.random.randint(10, 100, 40)
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(x, y, z, s=s)
Out[3]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x210eee5bfd0>
```

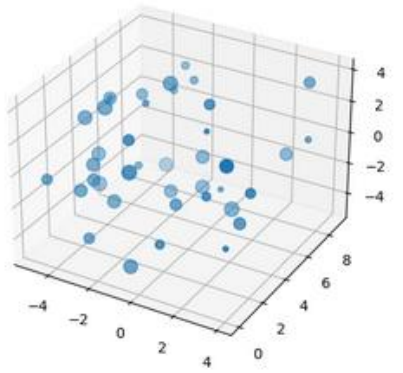
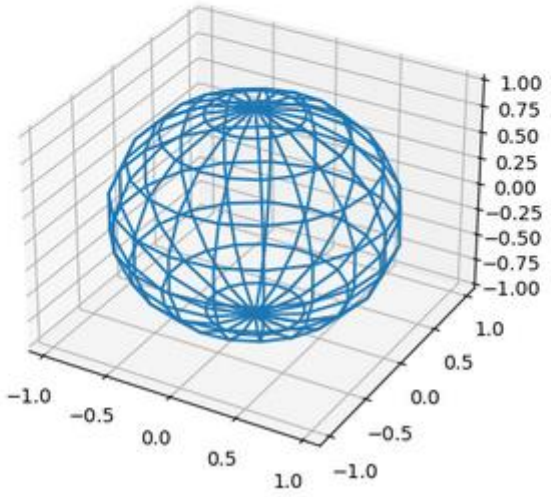


Рисунок 4 – Пример 1

Каркасная поверхность

In [4]:
```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(x, y, z)
```

Out[4]:
```
<mpl_toolkits.mplot3d.art3d.Line3DCollection at 0x210eead2f70>
```



Поверхность

In [5]:
```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='inferno')
```

Out[5]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x210eec4a5e0>
```

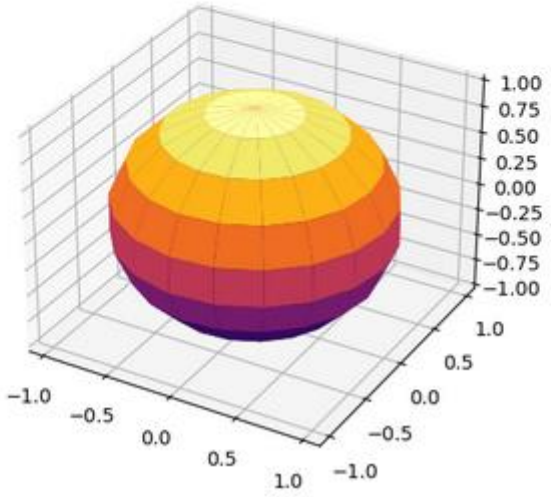


Рисунок 5 – Пример 2

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения трехмерного графика, условие которой предварительно необходимо согласовать с преподавателем.

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задач из области физики, экономики, математики, статистики и т. д.) требующей построения трехмерного графика, условие которой предварительно необходимо согласовать с преподавателем.

Построить график поверхности и каркасной поверхности для функции:
$f(x, y) = sin(x^2 + y^2) + 5 \cdot cos(x^2 + y^2) - 5$ при изменении значений переменных x и y от -10 до 20.

In [1]:
```
import numpy as np
from math import sqrt
from math import cos
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

Задаем значения, находим значение функции.

In [2]:
```
X = [float(i) for i in range(-10, 20)]
Y = [float(i) for i in range(-10, 20)]
X, Y = np.meshgrid(X, Y)
Z = (np.sqrt((X) ** 2 + (Y) ** 2) + 5 * np.cos(np.sqrt((X) ** 2 + (Y) ** 2)) - 5)
```

Строим график при помощи функции plot_surface().

In [3]:
```
fig = plt.figure(figsize=(5,5))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_title('Трехмерный график функции');
ax.set_xlabel("X", fontsize=15, labelpad=10)
ax.set_ylabel("Y", fontsize=15, labelpad=10)
ax.set_zlabel("Z", fontsize=15, labelpad=10)
plt.show()
```

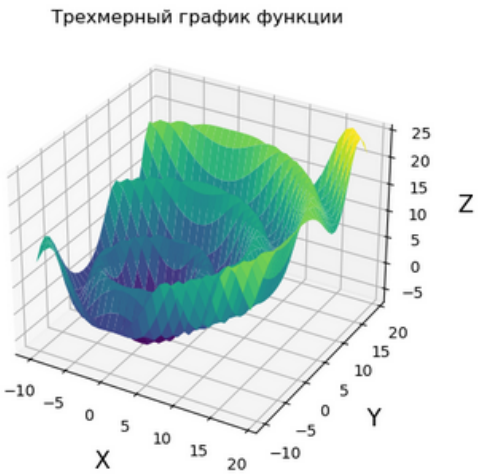


Рисунок 6 – Задание 1 (1)

График каркасной поверхности при помощи функции plot_wireframe().

```
In [4]: fig = plt.figure(figsize=(5,5))
        ax = fig.add_subplot(111, projection='3d')
        ax.plot_wireframe(X, Y, Z)
        ax.set_xlabel("X", fontsize=15, labelpad=10)
        ax.set_ylabel("Y", fontsize=15, labelpad=10)
        ax.set_zlabel("Z", fontsize=15, labelpad=10)
        plt.show()
```

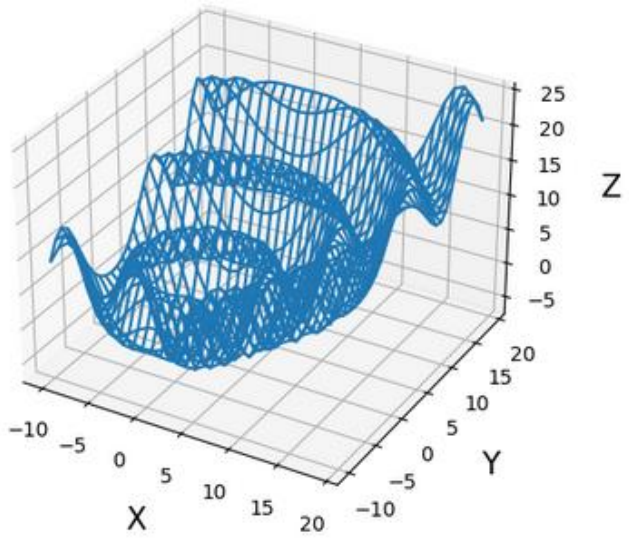


Рисунок 7 – Задание 1 (2)

Вывод: В результате выполнения работы было исследовано построение графиков при помощи модуля mplot3d Toolkit языка Python.

Контрольные вопросы:

1. Как выполнить построение линейного 3D-графика с помощью matplotlib?

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

Для построения линейного графика используется функция plot().

Axes3D.plot(self, xs, ys, *args, zdir='z', **kwargs)

xs: 1D-массив - x координаты.

ys: 1D-массив - y координаты.

zs: скалярное значение или 1D-массив - z координаты. Если передан скаляр, то он будет присвоен всем точкам графика.

zdir: {‘x’, ‘y’, ‘z’} - определяет ось, которая будет принята за z направление, значение по умолчанию: ‘z’.

**kwargs - дополнительные аргументы, аналогичные тем, что используются в функции plot() для построения двумерных графиков.

```
x = np.linspace(-np.pi, np.pi, 50)
y = x
z = np.cos(x)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, z, label='parametric curve')
```

2. Как выполнить построение точечного 3D-графика с помощью matplotlib?

Для построения точечного графика используется функция scatter().

Axes3D.scatter(self, xs, ys, zs=0, zdir=’z’, s=20, c=None, depthshade=True, *args, **kwargs)

xs, ys: массив - координаты точек по осям x и y.

zs: float или массив, optional - координаты точек по оси z. Если передан скаляр, то он будет присвоен всем точкам графика. Значение по умолчанию: 0.

zdir: {‘x’, ‘y’, ‘z’, ‘-x’, ‘-y’, ‘-z’}, optional - определяет ось, которая будет принята за z направление, значение по умолчанию: ‘z’

s: скаляр или массив, optional - размер маркера. Значение по умолчанию: 20.

c: color, массив, массив значений цвета, optional - цвет маркера. Возможные значения:

Строковое значение цвета для всех маркеров.

Массив строковых значений цвета.

Массив чисел, которые могут быть отображены в цвета через функции cmap и norm.

2D массив, элементами которого являются RGB или RGBA.

depthshade: bool, optional - затенение маркеров для придания эффекта глубины.

**kwargs - дополнительные аргументы, аналогичные тем, что используются в функции scatter() для построения двумерных графиков.

```
np.random.seed(123)
x = np.random.randint(-5, 5, 40)
y = np.random.randint(0, 10, 40)
z = np.random.randint(-5, 5, 40)
s = np.random.randint(10, 100, 20)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, s=s)
```

3. Как выполнить построение каркасной поверхности с помощью matplotlib?

Для построения каркасной поверхности используется функция plot_wireframe().

plot_wireframe(self, X, Y, Z, *args, **kwargs)

X, Y, Z: 2D-массивы - данные для построения поверхности.

rcount, ccount: int - максимальное количество элементов каркаса, которое будет использовано в каждом из направлений. Значение по умолчанию: 50.

rstride, cstride: int - параметры определяют величину шага, с которым будут браться элементы строки / столбца из переданных массивов. Параметры rstride, cstride и rcount, ccount являются взаимоисключающими.

**kwargs - дополнительные аргументы

```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(x, y, z)
ax.legend()
```

4. Как выполнить построение трехмерной поверхности с помощью matplotlib?

Для построения поверхности используйте функцию plot_surface().

plot_surface(self, X, Y, Z, *args, norm=None, vmin=None, vmax=None, lightsource=None, **kwargs)

X, Y, Z : 2D-массивы - данные для построения поверхности.

rcount, ccount : int - см. rcount, ccount в “Каркасная поверхность (https://devpractice.ru/matplotlib-lesson-5-1-mplot3d-toolkit/#p3)“.

rstride, cstride : int - см.rstride, cstride в “Каркасная поверхность (https://devpractice.ru/matplotlib-lesson-5-1-mplot3d-toolkit/#p3)“.

color: color - цвет для элементов поверхности.

cmap: Colormap - Colormap для элементов поверхности.

facecolors: массив элементов color - индивидуальный цвет для каждого элемента поверхности.

norm: Normalize - нормализация для colormap.

vmin, vmax: float - границы нормализации.

shade: bool - использование тени для facecolors. Значение по умолчанию: True.

lightsource: LightSource - объект класса LightSource – определяет источник света, используется, только если shade = True.

**kwargs - дополнительные аргументы

```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface()
```