

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

Кафедра инфокоммуникаций

**Отчет
по лабораторной работе №13
«Построение 3D графиков. Работа с mplot3d Toolkit»
по дисциплине:
«Введение в системы искусственного интеллекта»**

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Криворучко Ярослав Евгеньевич

_____ (подпись)

Проверил:
Воронкин Роман Александрович

_____ (подпись)

Ставрополь, 2022 г.

Цель работы: исследовать базовые возможности визуализации данных в трехмерном пространстве средствами библиотеки matplotlib языка программирования Python.

Ход работы:

```
Ввод [3]: #линейный график
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

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[3]: [<mpl_toolkits.mplot3d.art3d.Line3D at 0x1be50b8a5e0>]
```

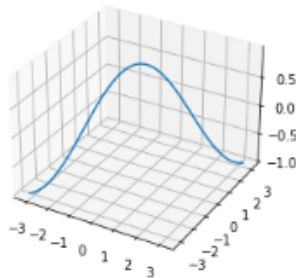


Рисунок 1 – Пример

```
Ввод [8]: #Поверхность
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
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')
ax.legend()

No handles with labels found to put in legend.

Out[8]: <matplotlib.legend.Legend at 0x1be50f462e0>
```

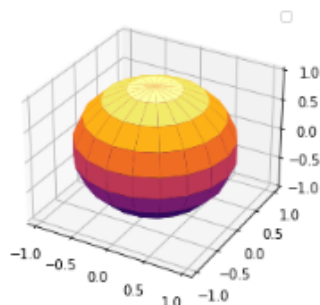


Рисунок 2 – Пример

```
Ввод [9]: #Каркасная поверхность
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
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()

No handles with labels found to put in legend.

Out[9]: <matplotlib.legend.Legend at 0x1be51699e20>
```

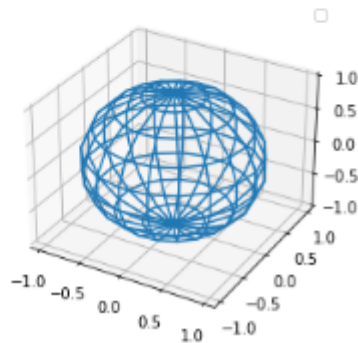


Рисунок 3 – Пример

Вывод: исследовал базовые возможности визуализации данных в трехмерном пространстве средствами библиотеки matplotlib языка программирования Python.

Ответы на вопросы:

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

Для построения линейного графика используется функция `plot()`.

- `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.

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

Для построения точечного графика используется функция `scatter()`.

- `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:

- o Строковое значение цвета для всех маркеров.
- o Массив строковых значений цвета.

- о Массив чисел, которые могут быть отображены в цвета через функции `cm` и `norm`.

- о 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.

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, определяемые `Line3DCollection`

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, lightsources=None, **kwargs)
```

- `X, Y, Z` : 2D-массивы - данные для построения поверхности.
- `rcount, ccount : int` - см. `rcount, ccount` в “Каркасная поверхность”

- `rstride, cstride : int` - см.`rstride, cstride` в “Каркасная поверхность”
- `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, определяемые `Poly3DCollection`