

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет о лабораторной работе №6 по дисциплине технологии
распознавания образов**

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

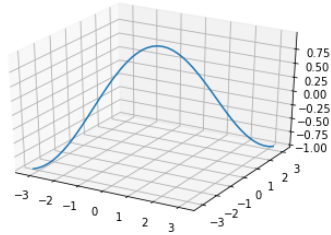
1. примеры из методических указаний

```
In [3]: import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

Линейный график

```
In [4]: 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')
```

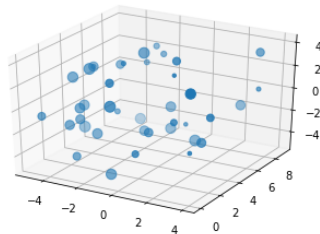
Out[4]: [



Точечный график

```
In [6]: 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)
```

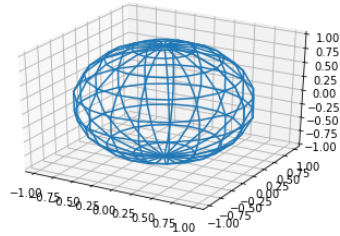
Out[6]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x4a7c112b20>



Каркасная поверхность

```
In [9]: 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)
```

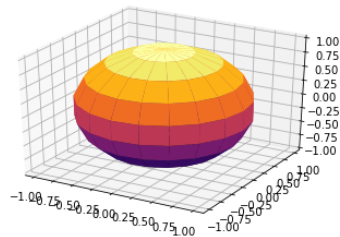
```
Out[9]: <mpl_toolkits.mplot3d.art3d.Line3DCollection at 0x4a7c8020a0>
```



Поверхность

```
In [10]: 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[10]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x4a7c81fa60>
```



2. Построение собственного 3D графика.

```
Ввод [10]: 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
```

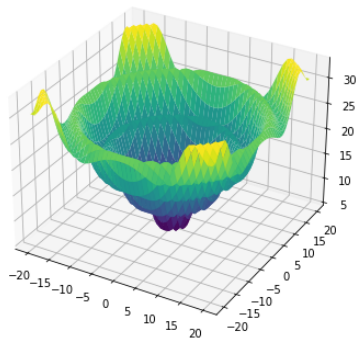
```
Ввод [33]: x = [float(i) for i in range(-20, 21)]
y = [float(i) for i in range(-20, 21)]
X, Y = np.meshgrid(x, y)
print(X)
print(Y)
Z = (np.sqrt((X) ** 2 + (Y) ** 2) + 3 * np.cos(np.sqrt((X) ** 2 + (Y) ** 2)) + 5)
print(Z)

[[-20. -19. -18. ... 18. 19. 20.]
 [-20. -19. -18. ... 18. 19. 20.]
 [-20. -19. -18. ... 18. 19. 20.]
 ...
 [-20. -19. -18. ... 18. 19. 20.]
 [-20. -19. -18. ... 18. 19. 20.]
 [-20. -19. -18. ... 18. 19. 20.]]
[[-20. -20. -20. ... -20. -20. -20.]
 [-19. -19. -19. ... -19. -19. -19.]
 [-18. -18. -18. ... -18. -18. -18.]
 ...
 [ 18. 18. 18. ... 18. 18. 18.]
 [ 19. 19. 19. ... 19. 19. 19.]
 [ 20. 20. 20. ... 20. 20. 20.]]
[[30.28441937 30.26887667 31.30033451 ... 31.30033451 30.26887667
 30.28441937]
 [30.26887667 31.37280281 32.69177745 ... 32.69177745 31.37280281
 30.26887667]
 [31.30033451 32.69177745 33.30060847 ... 33.30060847 32.69177745
 31.30033451]
 ...
 [31.30033451 32.69177745 33.30060847 ... 33.30060847 32.69177745
 31.30033451]
 [30.26887667 31.37280281 32.69177745 ... 32.69177745 31.37280281
 30.26887667]
 [30.28441937 30.26887667 31.30033451 ... 31.30033451 30.26887667
 30.28441937]]
```

```
Ввод [34]: fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(111, projection='3d')

# Plot a 3D surface
ax.plot_surface(X, Y, Z, cmap='viridis')

plt.show()
```



```

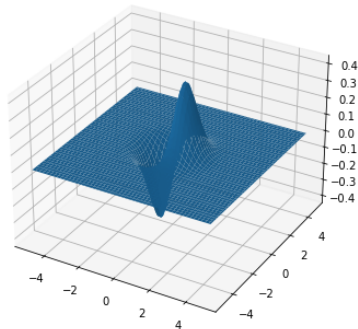
Ввод [39]: x = np.arange(-5,5,0.1)
y = np.arange(-5,5,0.1)
X,Y = np.meshgrid(x,y)
Z = X*np.exp(-X**2 - Y**2)

fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(111, projection='3d')

# Plot a 3D surface
ax.plot_surface(X, Y, Z)

plt.show()

```



Ввод []:

3. Ответы на 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.

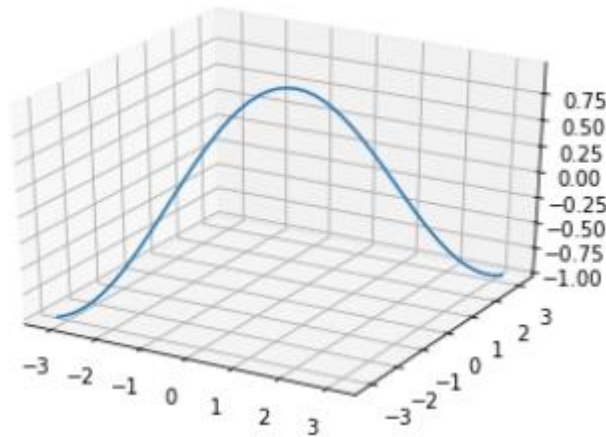
```
x = np.linspace(-np.pi, np.pi, 50)
```

```
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.

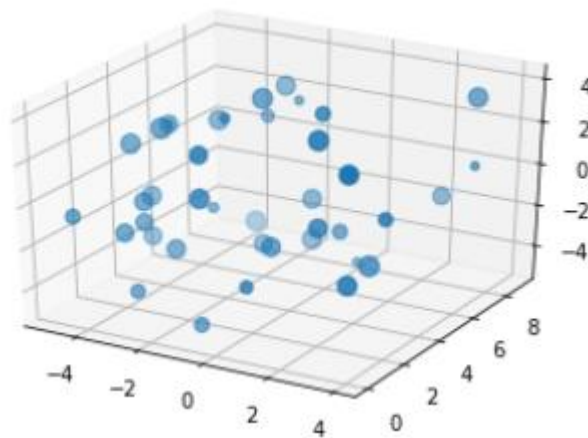
Массив чисел, которые могут быть отображены в цвета через функции `star` и `norm`.

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, scount: 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, scount 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)
```

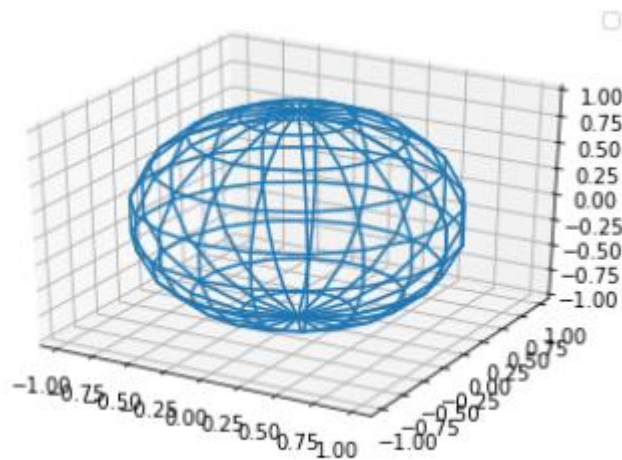
```
z = np.cos(v)
```

```
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(x, y, z, cmap='inferno')
```

```
ax.legend()
```

