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

Отчет по лабораторной работе №6 по дисциплине:

Технологии распознавания образов

Выполнила:
студент группы ПИЖ-б-о-20-1
Лазарева Дарья Олеговна

Проверил:
доцент кафедры инфокоммуникаций
Романкин Р.А.

Ставрополь, 2022 г.

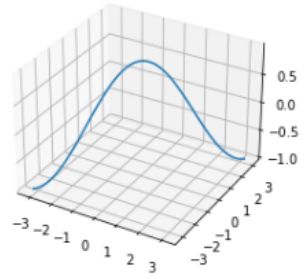
ВЫПОЛНЕНИЕ:

1. Линейный график

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

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')
```

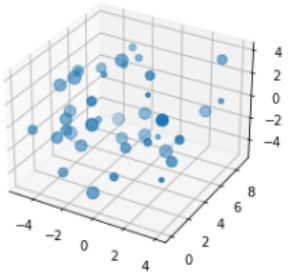
Out[2]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x1851f271430>]
```



2. 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)
```

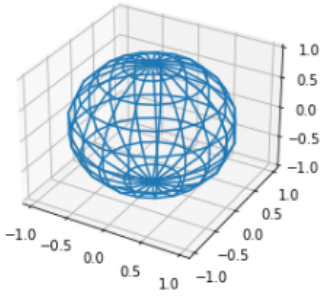
Out[3]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1851f37e400>
```



3. 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)
```

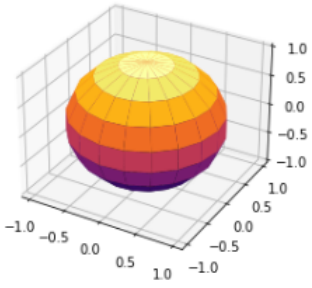
Out[4]: <mpl_toolkits.mplot3d.art3d.Line3DCollection at 0x1851f3f57f0>



4. 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 0x1851f478ee0>



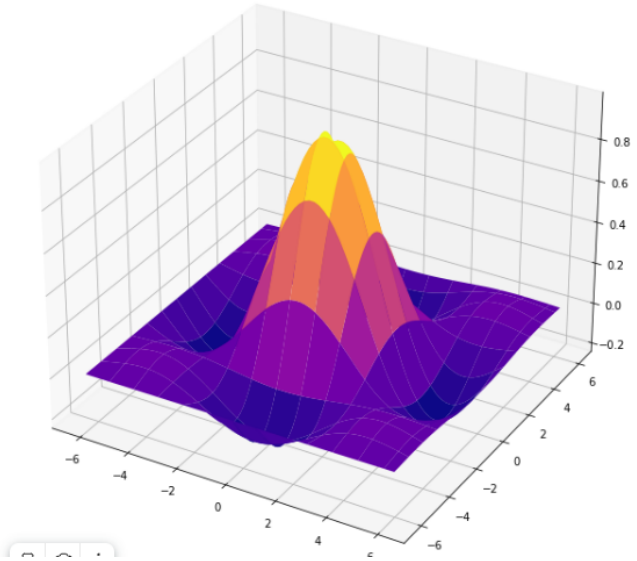
5. Выполнение индивидуального задания

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

        fig = plt.figure(figsize=(14, 8))
        ax_3d = Axes3D(fig)

        x = np.arange(-2*np.pi, 2*np.pi, 0.2)
        y = np.arange(-2*np.pi, 2*np.pi, 0.2)
        xgrid, ygrid = np.meshgrid(x, y)
        zgrid = np.sin(xgrid) * np.sin(ygrid) / (xgrid * ygrid)
        ax_3d.plot_surface(xgrid, ygrid, zgrid, rstride=5, cstride=5, cmap='plasma')

Out[1]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x172cf895f10>
```



Контрольные вопросы:

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

Matplotlib позволяет строить 3D графики. Импортируем необходимые модули для работы с 3D:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

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:
- 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.

3. Как выполнить построение каркасной поверхности с помощью matplotlib?

Для построения каркасной поверхности используется функция plot_wireframe().

- 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().

- 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