МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Отчёт по практическому занятию №3.6

«Построение 3D графиков. Работа с mplot3d Toolkit»

по дисциплине «Теории распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1
Коновалова В. Н. « »_____20__г.
Подпись студента _____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).

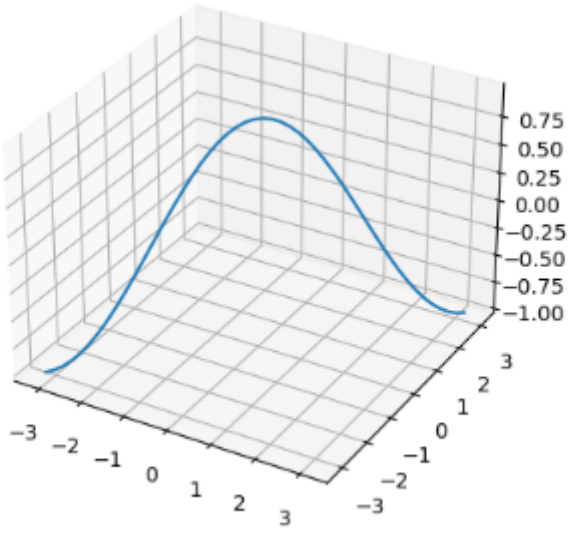
2. Проработать примеры лабораторной работы.

Примеры

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

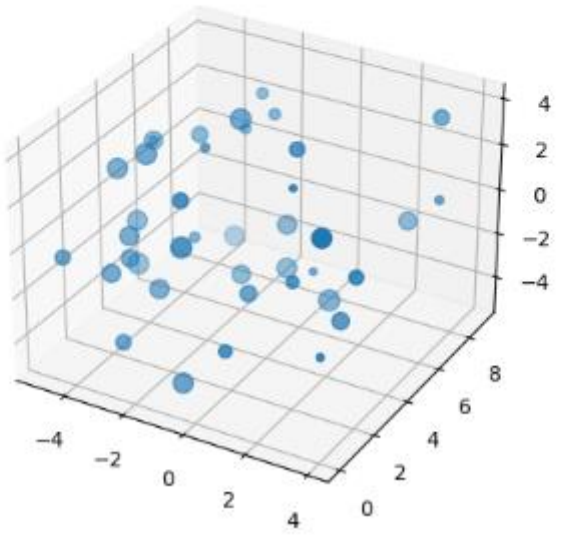
Линейный график

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');
```



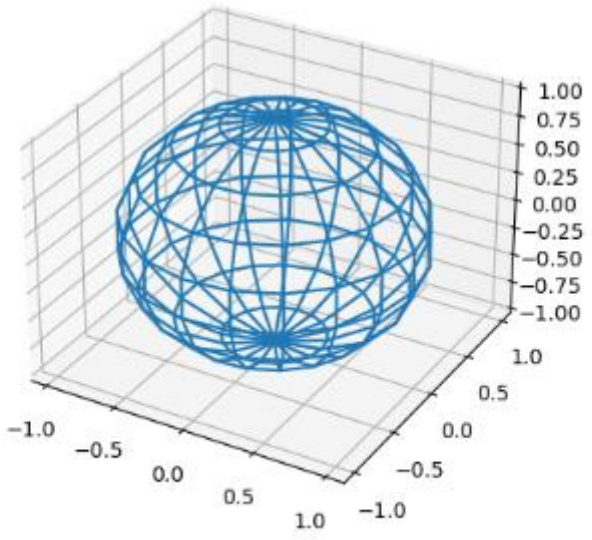
Точечный график

In [7]:
```
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);
```



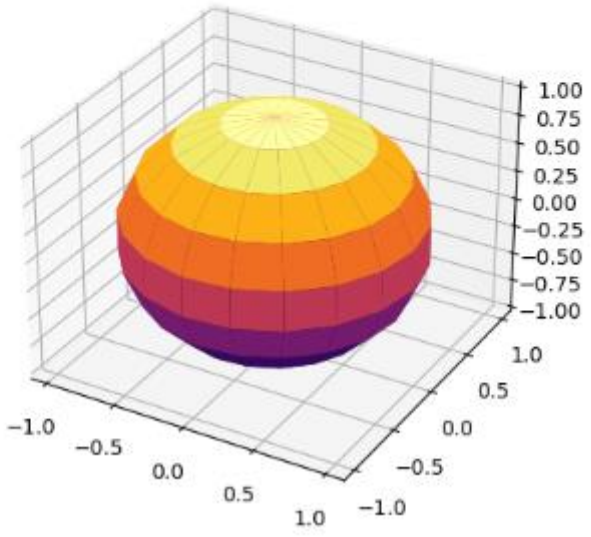
Каркасная поверхность

In [12]:
```
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);
```



Поверхность

In [14]:
```
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');
```



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.

Выполнить решение задачи из области математики о нахождении уравения плоскости по трём точкам в трехмерном простанстве и строит трехмерный график.

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

Зададим три точки в пространстве:

In [2]:
```
p1 = input("Введите координаты первой точки через пробел: ")

p2 = input("Введите координаты второй точки через пробел: ")

p3 = input("Введите координаты третьей точки через пробел: ")

P1 = np.array(list(map(float, p1.split())))
P2 = np.array(list(map(float, p2.split())))
P3 = np.array(list(map(float, p3.split())))
```

```
Введите координаты первой точки через пробел: 4 9 2
Введите координаты второй точки через пробел: 1 9 3
Введите координаты третьей точки через пробел: 9 3 7
```

Далее, мы можем вычислить векторы u и v, которые соответствуют сторонам треугольника, образованного тремя точками:

In [3]:
```
u = P2 - P1
v = P3 - P1
```

Затем, мы можем вычислить нормаль к плоскости, которая проходит через эти три точки, используя векторное произведение векторов u и v:

In [4]:
```
n = np.cross(u, v)
```

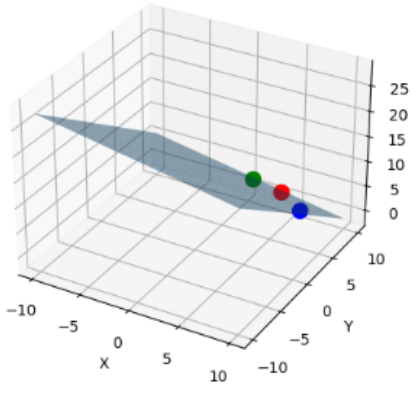
Иссползуем нормальный вектор n для записи уравнения плоскости в точечной форме:

In [5]:
```
d = -P1.dot(n)
xx, yy = np.meshgrid(range(-10, 11), range(-10, 11))

if np.isclose(n[2], 0):
    zz = np.full_like(xx, np.nan)
else:
    zz = (-n[0]*xx - n[1]*yy - d) / n[2]
```

Теперь мы можем построить трехмерный график, используя метод scatter():

In [6]:
```
# Установка параметров для построения графика
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
# Построение графика
ax.scatter(P1[0], P1[1], P1[2], color='red', s=100)
ax.scatter(P2[0], P2[1], P2[2], color='green', s=100)
ax.scatter(P3[0], P3[1], P3[2], color='blue', s=100)
ax.plot_surface(xx, yy, zz, alpha=0.5);
```



Вопросы для защиты работы

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

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, определяемые *Line3DCollection*(https://matplotlib.org/api/_as_gen/mpl_toolkits.mplot3d.art3d.Line3DCollection.html#mpl_toolkits.mplot3d.art3d.Line3DCollection).

```
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, определяемые *Poly3DCollection*(https://matplotlib.org/api/_as_gen/mpl_toolkits.mplot3d.art3d.Poly3DCollection.html#mpl_toolkits.mplot3d.art3d.Poly3DCollection).

```
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()
```