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

Кафедра инфокоммуникаций

Анализ данных

Отчет по лабораторной работе №3.6

Тема: «Построение 3D графиков. Работа с mplot3d Toolkit»

Выполнил студент группы

ИВТ-б-о-21-1

Богдан А. А. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций, старший преподаватель
Воронкин Р.А.

_____
(подпись)

Ставрополь 2023

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

Ход работы:

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

Рисунок 1.1 – Созданный репозиторий

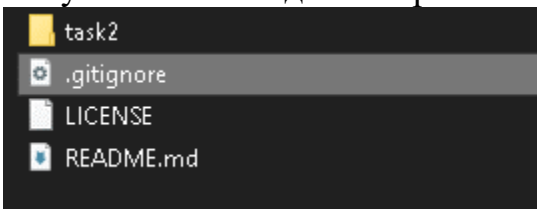


Рисунок 1.2 – Изменения в .gitignor

```
c:\Users\Admin\Desktop\git>cd /d c:\users\admin\desktop\git\Python22-3.6

c:\Users\Admin\Desktop\git\Python22-3.6>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/Python22-3.6/.git/hooks]

c:\Users\Admin\Desktop\git\Python22-3.6>
```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Проработал примеры лабораторной работы.

```
In [8]:
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');
```

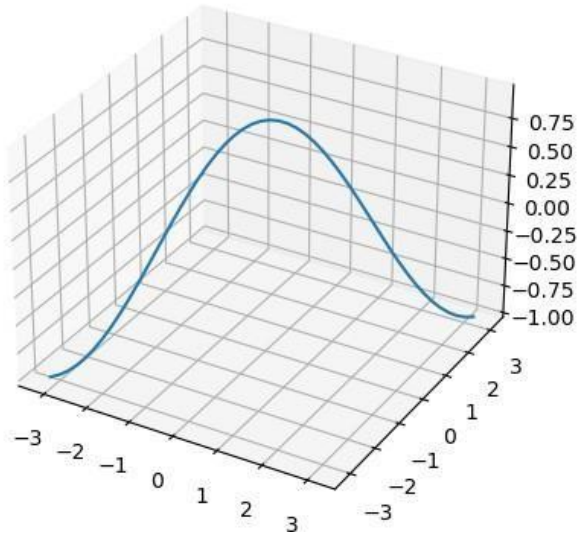


Рисунок 2 – Проработка примеров лабораторной работы

3. 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.

Зададим точки в пространстве:

In [257]:
```
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))
```

Посторим график

In [258]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

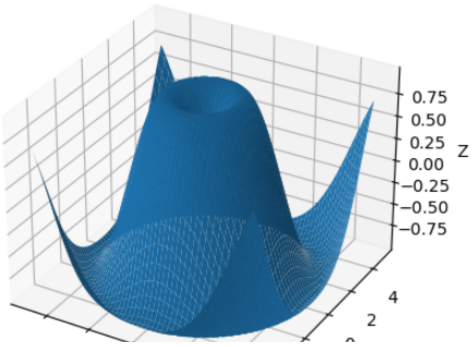


Рисунок 3 – Решение индивидуального задания

Вывод: в результате выполнения лабораторной работы были получены практические навыки и теоретические сведения необходимые для работы с базовыми возможностями визуализации данных в трехмерном пространстве средствами библиотеки matplotlib языка программирования Python.

Ответы на контрольные вопросы:

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

Линейный график

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?

Точечный график

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?

Каркасная поверхность

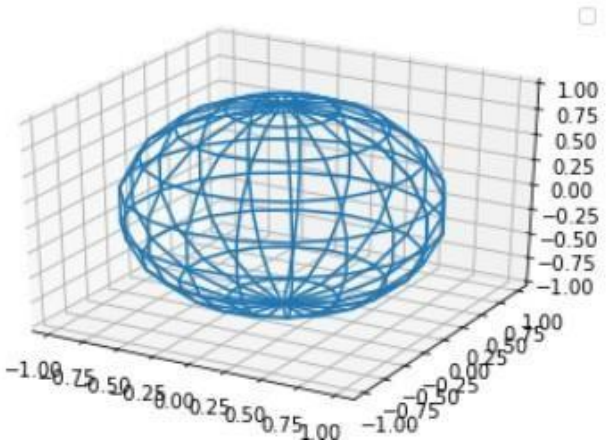
Для построения каркасной поверхности используется функция *plot_wireframe()*.

```
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?

Поверхность

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()
```