1 Введение

В этом документе объясняется, как создавать интерактивные слайдеры в Matplotlib для управления параметрами графиков. В качестве примера рассмотрим синусоидальную функцию с четырьмя параметрами, регулируемыми слайдерами.

2 Код с пояснениями

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
```

Импортируем необходимые библиотеки: NumPy для вычислений, Matplotlib для визуализации и модуль widgets для создания интерактивных элементов.

2.1 Исходные данные

```
x = np.linspace(0, 10, 100)

a = 0
b = 1
c = 1
d = 0

y = a + b * np.sin(c * x + d)
```

Создаем массив x от 0 до 10 и вычисляем значения функции $y = a + b \cdot \sin(c \cdot x + d)$ с начальными параметрами.

2.2 Создание графика

```
fig, ax = plt.subplots()
plt.subplots_adjust(bottom=0.5)
line, = ax.plot(x, y)
```

Создаем фигуру и оси, оставляем место внизу для слайдеров (bottom=0.5), рисуем начальный график.

2.3 Создание слайдеров

```
ax_a = plt.axes([0.1, 0.25, 0.65, 0.03])
a_slider = Slider(ax_a, 'a', 0.1, 10.0, valinit=a)
```

Для каждого параметра (a, b, c, d):

- Создаем область для слайдера с координатами [left, bottom, width, height]

- Создаем сам слайдер с указанием диапазона и начального значения

2.4 Функция обновления

```
def update(val):
    a = a_slider.val
    b = b_slider.val
    c = c_slider.val
    d = d_slider.val

    data = a + b * np.sin(c * x + d)
    line.set_ydata(data)
    fig.canvas.draw_idle()
```

Эта функция вызывается при изменении любого слайдера:

- Получает текущие значения всех слайдеров
- Пересчитывает данные графика
- Обновляет линию графика
- Перерисовывает фигуру

2.5 Привязка событий

```
a_slider.on_changed(update)
b_slider.on_changed(update)
c_slider.on_changed(update)
d_slider.on_changed(update)
```

Привязываем функцию update к событиям изменения каждого слайдера.

3 Построение 3d графиков в matplotlib

3.1 Создание сетки данных

```
X = np.linspace(-1, 1, 1000)
Z = np.linspace(-1, 1, 1000)
X, Z = np.meshgrid(X, Z)
```

- `np.linspace` создаёт равномерно распределённые точки в заданном диапазоне
- `np.meshgrid` создаёт прямоугольную сетку из двух одномерных массивов

3.2 Определение поверхности

```
Y = np.sqrt(1 - X ** 2)
Y2 = -Y
```

Здесь мы задаём уравнение поверхности. В данном случае это уравнение полусферы:

$$y = \pm\sqrt{1 - x^2}$$

Мы создаём две поверхности (Y и Y2) для верхней и нижней полусфер.

3.3 Создание 3D-графика

```
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
```

Ключевой момент - указание `projection='3d'`, которое создаёт трёхмерные оси.

3.4 Визуализация поверхностей

```
surf = ax.plot_surface(X, Y, Z, cmap='viridis')
surf = ax.plot_surface(X, Y2, Z, cmap='viridis')
```

Функция `plot_surface` принимает:

- X, Y, Z - координаты точек
- cmap - цветовую карту для визуализации

3.5 Настройка осей и заголовка

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

Эти команды добавляют подписи к осям

3.6 Отображение графика

```
plt.show()
```

Эта команда выводит интерактивное окно с графиком, который можно вращать и масштабировать.

4 Дополнительные возможности

1. Изменение цветовой карты (параметр `cmap`):

   - `'viridis'` (по умолчанию)
   - `'plasma'`, `'magma'`, `'inferno'`
   - `'coolwarm'`, `'rainbow'`

2. Регулировка прозрачности:

```
ax.plot_surface(X, Y, Z, alpha=0.5)
```