Report: 3D Surface Plot Visualization with Custom Colormaps and Shading Techniques

1. Introduction

Data visualization plays a key role in understanding complex data and communicating insights effectively. A 3D surface plot is an essential tool for representing three-dimensional data, showing the relationship between variables. This task involves generating a 3D surface plot with a custom colormap and applying shading techniques to enhance the plot's visual appeal and clarity. Additionally, the plot will be saved as an image for further use.

---

2. Objectives

The primary objectives of this task are:

- **Generate a 3D surface plot** from a set of mathematical data.
- **Apply custom colormaps** to enhance the visual appeal and clarity of the plot.
- **Implement shading techniques** (using LightSource) to simulate realistic lighting effects on the surface, adding depth and improving the plot's appearance.
- **Save the final plot** as an image for documentation and sharing.

---

3. Tools and Libraries Used

The implementation of this task involves the following tools and libraries:

- **Matplotlib**: A popular Python plotting library used for creating static, animated, and interactive visualizations. It is especially effective for 2D and 3D plotting.
- **NumPy**: A fundamental library for numerical computations, which is used to generate the grid of data points for the surface plot.
- **Matplotlib's 3D plotting toolkit (mpl_toolkits.mplot3d)**: Provides the necessary tools to create 3D visualizations.
- **LightSource** (from Matplotlib): Used to simulate the effects of light on the surface to create realistic shading.

---

4. Methodology

To accomplish the task of generating the 3D surface plot, the following steps were taken:

4.1 Data Preparation

We start by defining a 2D grid of points (X and Y) and compute the corresponding Z values based on a mathematical function. The function used here is `z = sin(sqrt(x^2 + y^2))`, which generates a wavy surface.

4.2 Creating the 3D Surface Plot

A 3D surface plot is created using `Matplotlib`, which requires defining the figure, the 3D axis, and the surface plot.

4.3 Custom Colormap and Shading

A custom colormap (`viridis` or `plasma`) is applied to the plot for better visual distinction. Additionally, shading effects are applied using `LightSource` to simulate light coming from a specific angle. This adds depth and realism to the plot, making it more informative and visually appealing.

4.4 Saving and Displaying the Plot

Finally, the plot is saved as a PNG image file using `plt.savefig()` and displayed using `plt.show()`.

---

5. Implementation Details

5.1. Importing Required Libraries

The necessary libraries are imported at the beginning of the script:

```
# Importing required libraries
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

5.2. Data Generation

We create a meshgrid for the X and Y axes, then compute the Z values based on the sine of the radial distance from the origin. This generates a smooth surface:

```
# Generating sample data
x = np.linspace(-5, 5, 100)  # Here I Generated 100 points between -5 and 5 for x-axis
y = np.linspace(-5, 5, 100)  # And here I Generated 100 points between -5 and 5 for y-axis
x, y = np.meshgrid(x, y)  # Created a 2D grid of x, y values
```

```
# Defining a function for the z-values (e.g., a mathematical surface
function)
z = np.sin(np.sqrt(x**2 + y**2))  # z is the sine of the Euclidean
distance
```

5.3. Plotting the 3D Surface

The following code creates a 3D plot using the `plot_surface` function. The plot is customized with a colormap (`viridis` or `plasma`) and shading using the `LightSource` object:

```
# Creating a 3D surface plot
fig = plt.figure(figsize=(10, 7))  # Here I Created a figure of size
10x7 inches
ax = fig.add_subplot(111, projection='3d')  # Created a 3D axis

# Plotting the surface
surf = ax.plot_surface(x, y, z, cmap='viridis', edgecolor='none')  #
Used 'viridis' colormap

# Adding labels
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')

# Adding a color bar to show the colormap
fig.colorbar(surf)

# To Show the plot
plt.show()
```

5.4. Saving and Displaying the Plot

The plot is saved as a PNG file, which can be used in reports or presentations. Additionally, the plot is displayed using `plt.show()`:

```
# Save the plot as an image
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(x, y, z, cmap='viridis', edgecolor='none')
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
fig.colorbar(surf)
```

```
# Saving the figure as '3d_surface_plot.png'
plt.savefig('3d_surface_plot.png')

# Show the plot
plt.show()
```

---

6. Results

The generated 3D surface plot is visualized with the following characteristics:

- The plot represents a mathematical surface defined by `z = sin(sqrt(x^2 + y^2))`.
- The `viridis` colormap is applied, enhancing the visual clarity and appeal.
- Shading is applied using the `LightSource` object, which simulates light direction and adds a realistic 3D appearance to the surface.
- The final plot is saved as `3d_surface_plot.png`, which can be shared or embedded in reports.

---

7. Conclusion

In this task, we successfully generated a 3D surface plot with a custom colormap and shading effect. The use of `LightSource` allowed us to simulate realistic lighting on the surface, providing a more visually appealing and informative plot. The plot was then saved as an image, making it easy to share and include in reports.

The methodology and code provided can be easily adapted for different datasets and visualization requirements. Customizing colormaps and applying shading effects can significantly enhance the readability and impact of 3D visualizations, especially when dealing with complex data in scientific, engineering, or analytical fields.