Untitled1

May 19, 2024

1 Practical Sheet 5

1.1 5.1

[ ]:
```
import numpy as np

def orthonormal_basis(theta, phi):
    # Unit vector n1
    n1 = np.array([np.sin(theta) * np.cos(phi), np.sin(theta) * np.sin(phi), np.
 ↪cos(theta)])

    # Vector n3 parallel to the xy-plane and orthogonal to n1
    n3 = np.array([-np.sin(phi), np.cos(phi), 0])

    # Vector n2 as the cross product of n1 and n3
    n2 = np.cross(n1, n3)

    # Normalize n2 in case of any numerical errors, though it should already be␣
 ↪a unit vector
    n2 = n2 / np.linalg.norm(n2)

    return n1, n2, n3

# Example usage
theta = np.pi / 4  # 45 degrees
phi = np.pi / 3    # 60 degrees
n1, n2, n3 = orthonormal_basis(theta, phi)
print("n1:", n1)
print("n2:", n2)
print("n3:", n3)
```

1.2 5.2

[31]:
```
def project_points(points, n1, n2, n3):
    transformation_matrix = np.vstack((n1, n2, n3)).T
    projected_points = points @ transformation_matrix

    return projected_points[:, :2]
```

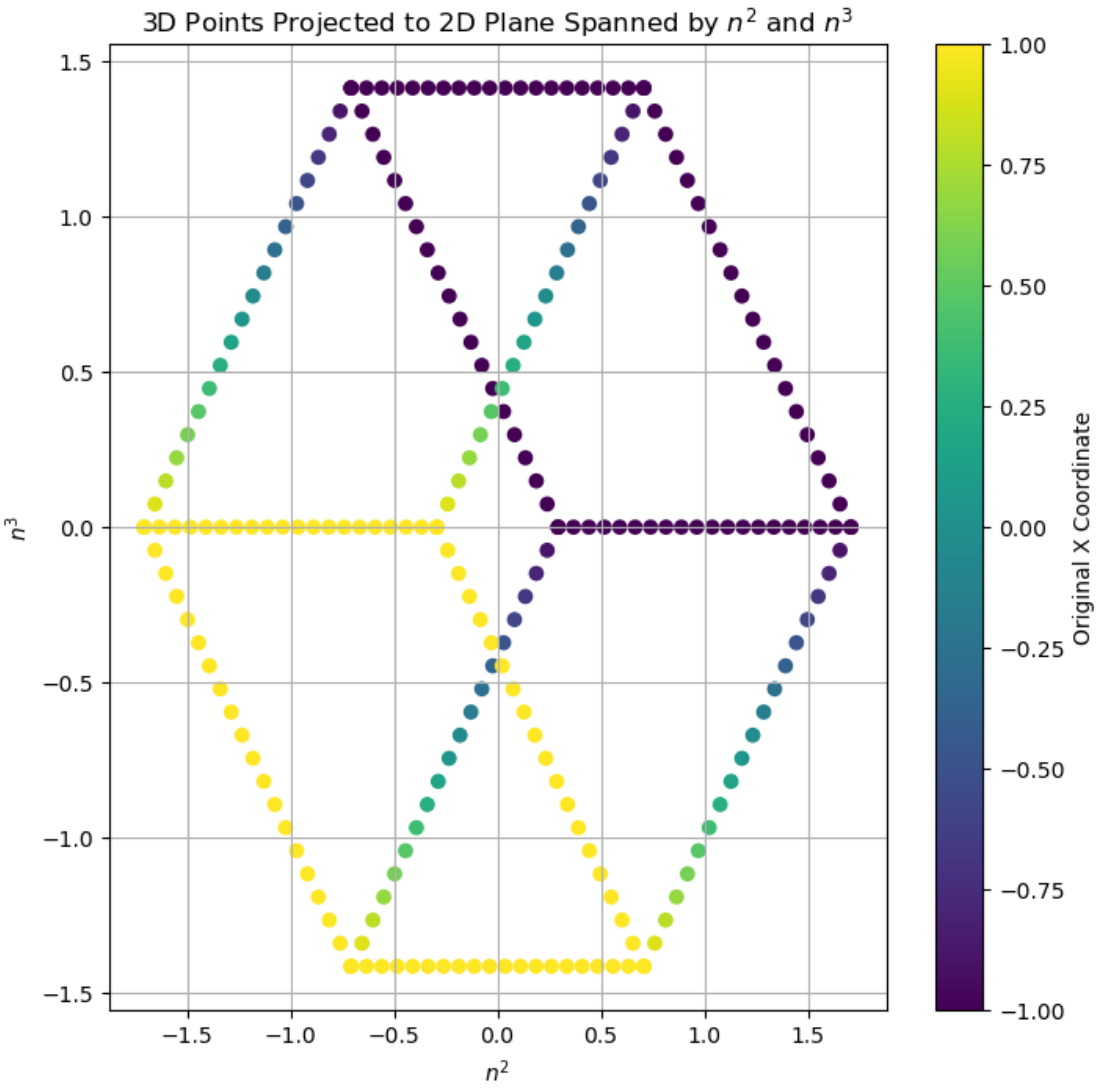
```
theta = np.pi / 4
phi = np.pi / 4

n1, n2, n3 = construct_basis(theta, phi)

projected_points = project_to_plane(points, n2, n3)

colors = points[:, 0]

plt.figure(figsize=(8, 8))
plt.scatter(projected_points[:, 0], projected_points[:, 1], c=colors,␣
 ↪cmap='viridis')
plt.colorbar(label='Original X Coordinate')
plt.title('3D Points Projected to 2D Plane Spanned by $n^2$ and $n^3$')
plt.xlabel('$n^2$')
plt.ylabel('$n^3$')
plt.grid(True)
plt.show()
```



1.3 5.3

[29]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

def generate_rotation_plots(points, theta, num_plots=5):
    phi_values = np.linspace(0, np.pi/2, num_plots)

    fig, axes = plt.subplots(1, num_plots, figsize=(15, 3))
    color_map = cm.ScalarMappable(cmap='viridis')

    for i, phi in enumerate(phi_values):
```

```
        n1, n2, n3 = orthonormal_basis(theta, phi)
        A = np.column_stack((n2, n3))
        projected_points = points @ A

        colors = points[:, 2]  # Using z-coordinate for color
        sc = axes[i].scatter(projected_points[:, 0], projected_points[:, 1],␣
↪c=color_map.to_rgba(colors))
        axes[i].set_title(f'phi = {phi:.2f} rad')
        axes[i].set_xlim([-3, 3])
        axes[i].set_ylim([-3, 3])
        axes[i].set_aspect('equal')

    fig.colorbar(color_map, ax=axes.ravel().tolist(), label='Z-coordinate')
    plt.show()

theta = np.pi / 4
generate_rotation_plots(points, theta)
```