Möbius Strip Modeling and Analysis using Python

Objective:

To model a Möbius strip using parametric equations in Python, compute its surface area and edge length numerically, and visualize the 3D surface.

---

1. Introduction to the Möbius Strip

A Möbius strip is a non-orientable surface with only one side and one boundary. It is formed by taking a rectangular strip, giving it a half-twist, and joining the ends. This project models the Möbius strip mathematically and computes its geometric properties.

---

2. Parametric Equations

Given:

- Radius (R): Distance from the center of the Möbius strip to the midline
- Width (w): Width of the strip
- Parameters:
  - $u \in [0, 2\pi]$
  - $v \in [-w/2, w/2]$

The 3D parametric equations are:

$$x(u,v) = \left(R + v \cdot \cos\left(\frac{u}{2}\right)\right) \cdot \cos(u)$$
$$y(u,v) = \left(R + v \cdot \cos\left(\frac{u}{2}\right)\right) \cdot \sin(u)$$
$$z(u,v) = v \cdot \sin\left(\frac{u}{2}\right)$$

These equations describe the Möbius surface with a half-twist.

---

3. Implementation in Python

A `MobiusStrip` class was implemented with the following methods:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps

class MobiusStrip:
    def __init__(self, R, w, n):
        # Radius of the strip center
        self.R = R
```

```python
        # Width of the strip
        self.w = w
        # Resolution for mesh generation
        self.n = n

    def compute_mesh(self):
        # Create parameter grids u and v
        u = np.linspace(0, 2 * np.pi, self.n)
        v = np.linspace(-self.w / 2, self.w / 2, self.n)
        u, v = np.meshgrid(u, v)

        # Parametric equations of the Mobius strip
        x = (self.R + v * np.cos(u / 2)) * np.cos(u)
        y = (self.R + v * np.cos(u / 2)) * np.sin(u)
        z = v * np.sin(u / 2)

        return x, y, z, u, v

    def compute_surface_area(self):
        # Compute mesh and partial derivatives
        x, y, z, u, v = self.compute_mesh()
        dx_u, dx_v = np.gradient(x)
        dy_u, dy_v = np.gradient(y)
        dz_u, dz_v = np.gradient(z)

        # Compute cross product of partial derivatives
        cross_x = dy_u * dz_v - dz_u * dy_v
        cross_y = dz_u * dx_v - dx_u * dz_v
        cross_z = dx_u * dy_v - dy_u * dx_v

        # Surface area element (magnitude of the cross product)
        dA = np.sqrt(cross_x**2 + cross_y**2 + cross_z**2)

        # Integrate using Simpson's rule
        area = simps(simps(dA, dx=v[0]), dx=u[:, 0])
        return area

    def compute_edge_length(self):
        # Edge curve along v = -w/2 and v = +w/2
        u = np.linspace(0, 2 * np.pi, self.n)
        v = self.w / 2

        # Parametric curve at the edge
        x = (self.R + v * np.cos(u / 2)) * np.cos(u)
        y = (self.R + v * np.cos(u / 2)) * np.sin(u)
        z = v * np.sin(u / 2)

        # Compute differential arc length
        dx = np.gradient(x)
        dy = np.gradient(y)
        dz = np.gradient(z)
        ds = np.sqrt(dx**2 + dy**2 + dz**2)

        # Integrate edge length
        length = np.sum(ds)
        return length

    def plot(self):
        # Generate mesh for plotting
        x, y, z, _, _ = self.compute_mesh()
        fig = plt.figure(figsize=(10, 6))
        ax = fig.add_subplot(111, projection='3d')

        # Plot the surface
        ax.plot_surface(x, y, z, cmap='viridis', edgecolor='none', alpha=0.8)
        ax.set_title('Mobius Strip')
        ax.set_xlabel('X')
```

```
        ax.set_ylabel('Y')
        ax.set_zlabel('Z')
        plt.tight_layout()
        plt.show()
```

---

4. Input Parameters

```
R = 1.0
w = 0.2
n = 200
```

Where:

- R = radius of the strip center
- w = width of the strip
- n = resolution (number of sampling points)

---

5. Mathematical calculations

i. Surface Area Calculation

We use the parametric surface equations:

We use the parametric surface equations:

$$x(u,v) = (R + v\cos(u/2))\cos(u)$$
$$y(u,v) = (R + v\cos(u/2))\sin(u)$$
$$z(u,v) = v\sin(u/2)$$

where:

- $u \in [0, 2\pi]$
- $v \in [-w/2, w/2] = [-0.1, 0.1]$

Surface Area Formula

The differential area element on a surface is:

$$dA = \left\| \frac{\partial \mathbf{r}}{\partial u} \times \frac{\partial \mathbf{r}}{\partial v} \right\| du\, dv$$

Where:

- $\mathbf{r}(u,v) = [x(u,v), y(u,v), z(u,v)]$
- $\frac{\partial \mathbf{r}}{\partial u}$ and $\frac{\partial \mathbf{r}}{\partial v}$ are computed numerically (with numpy.gradient)
- The norm of the cross product gives the infinitesimal area at each (u, v)

We numerically integrate this over the grid using the Simpson's Rule:

$$A \approx \int_0^{2\pi} \int_{-0.1}^{0.1} \left\| \frac{\partial \mathbf{r}}{\partial u} \times \frac{\partial \mathbf{r}}{\partial v} \right\| dv\, du$$

With n = 200 steps in both u and v, this yields:

- Final computed Surface Area ≈ 3.9886 × 10^{-5} units^2

ii. Edge Length Calculation

We consider the boundary curve of the strip at v = +w/2 = 0.1.

$$x(u) = (R + 0.1\cos(u/2))\cos(u)$$
$$y(u) = (R + 0.1\cos(u/2))\sin(u)$$
$$z(u) = 0.1\sin(u/2)$$

To compute the arc length:

$$L = \int_0^{2\pi} \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}\, du$$

This is done numerically by:

- Evaluating (x(u), y(u), z(u)) at n = 200 points
- Approximating the derivative using finite differences
- Summing the segment lengths

📊 Final Edge Length ≈ 6.3216 units

---

6. Results

- **Surface Area:** $\approx 3.9886 \times 10^{-5}$ units²
- **Edge Length:** ≈ 6.3216 units

These were computed using 200 sampling points in both u and v directions.

---

7. Visualization

A 3D plot of the Möbius strip was generated using `matplotlib.pyplot`'s `plot_surface()` function. The visualization confirms the expected twisted band topology.

---

8. Code Structure and Quality

- **Modular**: Code is organized into a class and functions
- **Clean**: Clear naming conventions, minimal repetition
- **Commented**: Key steps and formulas are explained inline

---

9. Challenges Faced

• Ensuring numerical stability and accuracy in gradient and integration calculations required careful use of vectorized operations and mesh resolution tuning.

• Correctly parameterizing the Möbius strip so the mesh wrapped seamlessly without artifacts was critical and required precise trigonometric expressions.

• Managing the edge curve length calculation involved handling parameter boundaries consistently to avoid distortions.

---

10. Conclusion

The Möbius strip was successfully modeled and analyzed using parametric equations. Numerical methods provided accurate estimations of geometric properties, and visualization demonstrated its topology.

Appendix: Sample Output Plot

- R = radius of the strip center = 1.0
- w = width of the strip = 0.2
- n = resolution (number of sampling points)=200