```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import brentq

def surface_point(theta, v, lam):
    """Équations géométriques de la surface de Tannery."""
    x = lam * 2 * np.sqrt(2) * np.sin(theta / 2)
    y = lam * np.sin(theta) * np.cos(v)
    z = lam * np.sin(theta) * np.sin(v)
    return x, y, z

def calculer_longueur_boucle(theta_max, v_base, direction_haut, lam, W=0.4,
n_pts=1000):
    """Calcule la longueur d'une boucle 3D fermée pour une altitude theta_max
donnée."""
    t = np.linspace(0, 2 * np.pi, n_pts)
    theta = theta_max * np.sin(t / 2)
    v = v_base + W * np.sin(t)

    x, y, z = surface_point(theta, v, lam)
    if direction_haut:
        x = -x

    dx = np.diff(x)
    dy = np.diff(y)
    dz = np.diff(z)
    return np.sum(np.sqrt(dx**2 + dy**2 + dz**2))

def generer_vraie_boucle(L_cible, v_base, direction_haut, lam, W=0.4,
n_pts=1000):
    """Trouve le theta_max parfait pour obtenir EXACTEMENT L_cible et génère la
boucle."""
    # Fonction objectif pour brentq : longueur_mesurée(theta_max) - L_cible = 0
    obj_func = lambda t_max: calculer_longueur_boucle(t_max, v_base,
direction_haut, lam, W, n_pts) - L_cible

    # Recherche du theta_max idéal entre 0 et pi
    theta_max_ideal = brentq(obj_func, 1e-5, np.pi, xtol=1e-12)

    # Génération finale de la courbe avec le theta_max optimal
    t = np.linspace(0, 2 * np.pi, n_pts)
    theta = theta_max_ideal * np.sin(t / 2)
    v = v_base + W * np.sin(t)

    x, y, z = surface_point(theta, v, lam)
    if direction_haut:
        x = -x

    return x, y, z

def tracer_sablier_goldbach_parfait():
    # Invariant total strict fixé à 98
    L_total = 98.0
    lam = (L_total / 2.0) / np.pi  # lam = 49 / pi

    fig = plt.figure(figsize=(13, 8))
    ax = fig.add_subplot(111, projection="3d")

    # 1. Tracé des poires de Tannery lisses en arrière-plan
    t_mesh = np.linspace(1e-4, np.pi, 60)
    v_mesh = np.linspace(0, 2 * np.pi, 60)
    TH, V = np.meshgrid(t_mesh, v_mesh)
```

```
    Xb, Yb, Zb = surface_point(TH, V, lam)
    ax.plot_surface(Xb, Yb, Zb, color="teal", alpha=0.05, linewidth=0,
antialiased=True)
    ax.plot_surface(-Xb, Yb, Zb, color="teal", alpha=0.05, linewidth=0,
antialiased=True)

    # Les 3 couples Goldbach pour 98
    decompositions = [(19, 79), (31, 67), (37, 61)]
    couleurs = ["#1f77b4", "#ff7f0e", "#2ca02c"]

    print("--- VÉRIFICATION ARITHMÉTIQUE DES HUIT FERMÉS DANS L'ESPACE ---")

    for i, ((p, q), col) in enumerate(zip(decompositions, couleurs)):
        # Angle de rotation distinct pour chaque paire de Goldbach
        v_base = (2 * np.pi * i) / 3

        # Génération de la boucle du haut (longueur arithmétique exacte = p)
        # Largeur W ajustable pour donner plus ou moins de "gonflant" aux 8
        xh, yh, zh = generer_vraie_boucle(p, v_base, direction_haut=True,
lam=lam, W=0.35)

        # Génération de la boucle du bas (longueur arithmétique exacte = q)
        xb, yb, zb = generer_vraie_boucle(q, v_base, direction_haut=False,
lam=lam, W=0.35)

        # Mesure de contrôle physique par intégration de segments 3D
        long_h = np.sum(np.sqrt(np.diff(xh)**2 + np.diff(yh)**2 +
np.diff(zh)**2))
        long_b = np.sum(np.sqrt(np.diff(xb)**2 + np.diff(yb)**2 +
np.diff(zb)**2))
        total_8 = long_h + long_b

        print(f"Paire {p} + {q} = 98 :")
        print(f"  -> Boucle Supérieure (p) : Longueur mesurée = {long_h:.4f}
(Cible = {p})")
        print(f"  -> Boucle Inférieure (q) : Longueur mesurée = {long_b:.4f}
(Cible = {q})")
        print(f"  -> TOTAL DU HUIT (8)     : {total_8:.4f} (Invariant Strict)")
        print(f"  -> Écart numérique       : {abs(total_8 - L_total):.2e}\n")

        # Tracé des trajectoires orbitales
        ax.plot(xh, yh, zh, color=col, linewidth=3, zorder=10)
        ax.plot(xb, yb, zb, color=col, linewidth=3, zorder=10)
        ax.plot([], [], [], color=col, linewidth=3, label=f"Huit {p} (haut) +
{q} (bas) = 98")

    # Point de contact central strict unique
    ax.scatter([0], [0], [0], color="black", s=120, marker="o", zorder=12,
label="Nœud (0,0,0)")

    # Graphisme
    ax.set_xlabel("X (Axe du Sablier)")
    ax.set_ylabel("Y")
    ax.set_zlabel("Z")
    ax.set_title("Vrais 8 fermés et asymétriques (Invariant de Tannery = 98)",
fontsize=12)
    ax.legend(loc="upper left", bbox_to_anchor=(1.02, 1.0))
    ax.view_init(elev=20, azim=115)

    # Cadrage
    lim = 45
    ax.set_box_aspect([2*lim, lim, lim])
    ax.set_xlim(-lim, lim)
    ax.set_ylim(-lim/2, lim/2)
```

```
    ax.set_zlim(-lim/2, lim/2)

    plt.tight_layout()
    plt.show()

tracer_sablier_goldbach_parfait()
```