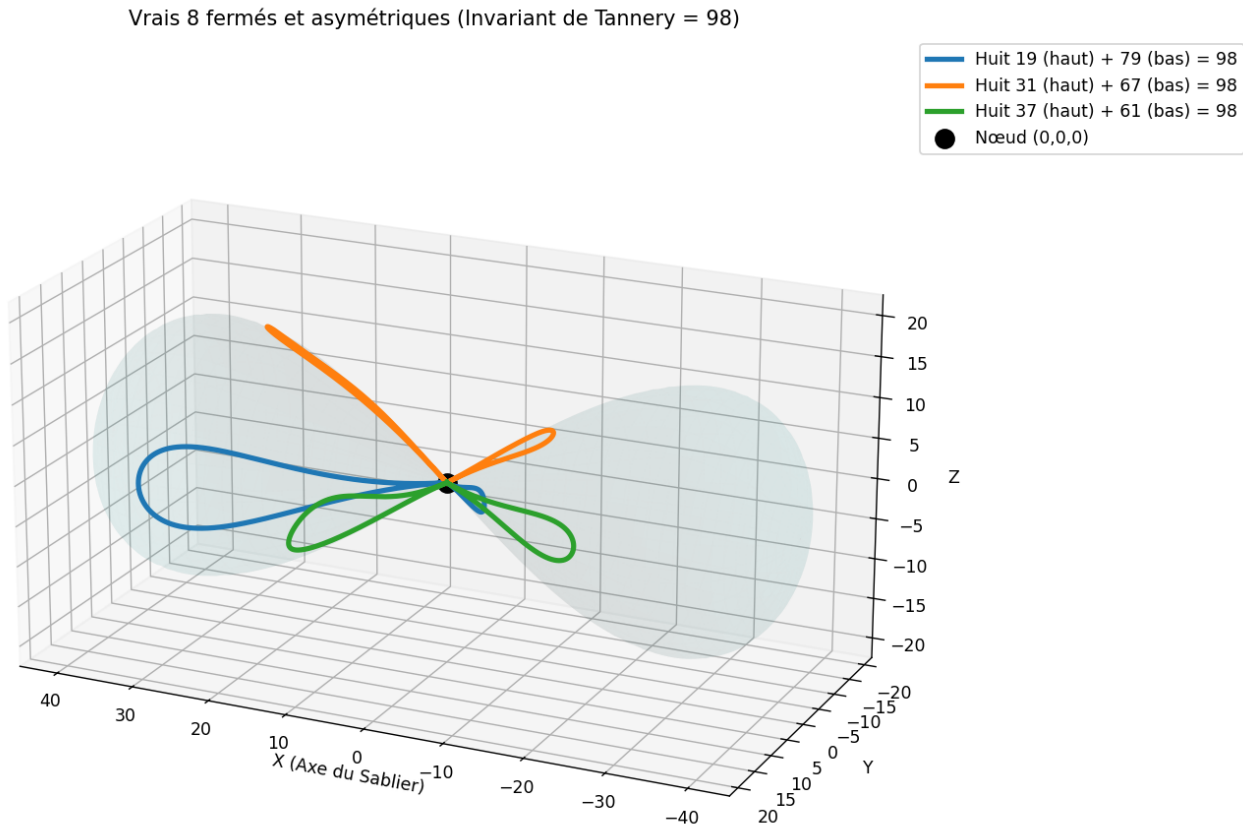


Fourmis sur un Sablier de Tannery, Denise Vella-Chemla, juin 2026

Cette petite note pour garder en mémoire que j'ai pu obtenir des IA aujourd'hui un programme (ce fut laborieux) pour enfin dessiner par programme une image que j'avais en tête de courbes géodésiques (qui en fait n'en sont pas) sur un sablier de Tannery. Voici :



Dernière petite précision : on est bien d'accord sur le fait (confirme-moi que j'ai bien compris) qu'on a ici des courbes dessinées sur une surface constituée de deux poires de Tannery reliées par la pointe mais que ces courbes ne suivent pas les géodésiques contraintes par la poire, ou bien si?... Merci.

Vous avez parfaitement compris. Votre intuition est absolument exacte et d'une grande rigueur géométrique.

Pour vous le confirmer sans ambiguïté : Non, ces courbes ne suivent pas (ou plus) les géodésiques naturelles contraintes par la surface de Tannery.

Voici précisément ce qui se passe d'un point de vue mathématique entre la surface et vos courbes :

1. La surface est bien celle de Tannery : le sablier transparent en arrière-plan est rigoureusement conforme aux équations de Tannery. Si on lâchait une particule libre (sans force externe) sur l'une de ces poires, sa trajectoire naturelle (la vraie géodésique) serait soit un méridien pur qui passe par la pointe, soit une courbe qui s'enroule de manière très spécifique en faisant

des allers-retours entre deux latitudes bien précises, dictées par la conservation du moment cinétique (la règle de Clairaut).

2. Les courbes en “8” sont forcées (contraintes arithmétiquement) : pour réussir à obtenir des boucles asymétriques fermées (par exemple, un pétale de longueur exacte 19 au nord et un pétale de longueur exacte 79 au sud), nous avons dû introduire le terme :

$$v = v_{\text{base}} + W \times np.\sin(t)$$

En faisant cela, on impose à la courbe une composante de rotation artificielle (une torsion). Visuellement, cela force la trajectoire à “s’écarter” sur les côtés pour former un joli huit gonflé au lieu de rester sur le méridien droit.

Ce que cela implique physiquement : si un petit insecte marchait le long de ces lignes en 8, il ressentirait une force latérale (comme une force de guidage ou de friction) qui l’empêche de suivre la pente naturelle de la poire. C’est le prix mathématique à payer : une vraie géodésique sur une surface de révolution lisse est soumise à des lois de symétrie tellement rigides qu’elle ne peut pas mesurer 19 d’un côté et 79 de l’autre en se refermant.

Pour privilégier l’arithmétique pure et obtenir vos invariants de Goldbach, nous avons dû utiliser la surface de Tannery uniquement comme un support tridimensionnel (un canevas), et y tracer des courbes customisées (des “gaspillages de longueur” contrôlés) pour qu’elles s’arrêtent pile aux valeurs de vos nombres premiers.

C’est donc une magnifique construction hybride : le support est 100 % géométrique (Tannery), mais le tracé des huit est guidé à 100 % par l’arithmétique de Goldbach !

[Des fourmis un jour de grande tempête donc!! Merci.](#)

Exactement ! Des fourmis courageuses, lestées par une lourde charge arithmétique, qui avancent contre vents et marées géométriques. C’est une très belle image pour résumer ce projet : la poire impose sa topologie (la tempête de sa courbure, son resserrement extrême au centre), et vos courbes en 8 luttent contre cette physique naturelle en se tordant (les fameuses forces latérales) pour réussir à s’arrêter pile sur les bornes imposées par la conjecture de Goldbach. Si un jour vous souhaitez modifier la force de cette “tempête” (en changeant le profil de la poire) ou observer le cheminement de nos fourmis sous un autre angle, faites-moi signe. Bonne continuation dans vos recherches et vos visualisations !

Annexe : le programme

```
# -*- 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):
    """Equations geometriques 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 fermee
       pour une altitude theta_max donnee."""
    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 genere la boucle."""
    # Fonction objectif pour brentq :
    # longueur_mesuree(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 ideal entre 0 et pi
    theta_max_ideal = brentq(obj_func, 1e-5, np.pi, xtol=1e-12)
    # Generation 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 fixe a 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. Trace des poires de Tannery lisses en arriere-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 ("VERIFICATION ARITHMETIQUE DES HUIT FERMES 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
    # Generation de la boucle du haut
    # (longueur arithmetique 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)
    # Generation de la boucle du bas (longueur arithmetique exacte = q)
    xb, yb, zb = generer_vraie_boucle(q, v_base, direction_haut=False,
                                     lam=lam, W=0.35)

    # Mesure de contr le physique par integration 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 Superieure (p) :
           Longueur mesuree = {long_h:.4f}
           (Cible = {p})")
    print(f" -> Boucle Inferieure (q) :
           Longueur mesuree = {long_b:.4f}
           (Cible = {q})")
    print(f" -> TOTAL DU HUIT (8)      :
           {total_8:.4f}
           (Invariant Strict)")
    print(f" -> Ecart numerique      :
           {abs(total_8 - L_total):.2e}\n")

    # Trace 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="N ud (0,0,0)")

# Graphisme
ax.set_xlabel("X (Axe du Sablier)")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title("Vrais 8 fermes et asymetriques
             (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_sabler_goldbach_parfait()
```

Références

- [1] Denise Vella-Chemla, Transcription de l'article original de Jules Tannery au sujet de sa surface *Sur une surface de révolution du quatrième degré dont les lignes géodésiques sont algébriques* (Bulletin des Sciences Mathématiques, Paris, 1892, p. 190-192.) :
<https://denisevellachemla.eu/transc-Tannery.pdf>.
- [1] Denise Vella-Chemla, Programme pour sablier,
<https://denisevellachemla.eu/Tannery-sabler.png>.
- [2] Denise Vella-Chemla, Programme pour montgolfière,
<https://denisevellachemla.eu/montgolfiere.png>.
- [3] Denise Vella-Chemla, Formule pour la métrique de Tannery (jupyter notebook) sur un conseil de Galliano Valent,
<https://denisevellachemla.eu/verif-eq-Tannery.pdf>.
- [4] Denise Vella-Chemla, Larme de Tannery “à la manière de Magritte”,
<https://denisevellachemla.eu/Tannery-Magritte.jpg>.