Animația unei Cuadrice Dublu Riglate

Cocu Matei-Iulian și Lăutaru Bianca-Maria

May 29, 2025

Documentație pentru generarea și vizualizarea unui hiperboloid cu o pânză

Contents

1 Introducere

Acest document descrie procesul de generare a unei animații tridimensionale pentru o cuadrică dublu riglată, specific un hiperboloid cu o pânză, utilizând limbajul de programare Python și biblioteca Matplotlib. Scopul este de a vizualiza cele două familii de drepte (generatoare) care compun suprafața cuadricei și de a observa structura acesteia în timpul rotației. Codul Python implementează calculul parametric al acestor drepte și orchestrarea animației.

2 Cuadrice Dublu Riglate - Aspecte Teoretice

Definiție

O **suprafață riglată** este o suprafață generată prin mișcarea unei drepte în spațiu. Această dreaptă mobilă se numește **generatoare** a suprafeței. O **cuadrică dublu riglată** este o suprafață cuadrică (definită de o ecuație de gradul al doilea) care conține nu una, ci *două* familii distincte de generatoare. Prin fiecare punct al unei astfel de suprafețe trec exact două drepte distincte care sunt în întregime conținute în suprafață, câte una din fiecare familie.

Principalele tipuri de cuadrice dublu riglate sunt:

- **Hiperboloidul cu o pânză:** Este suprafața cuadrică definită canonic de ecuația:
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$
Aceasta este cuadrica implementată în codul Python furnizat.
- **Paraboloidul hiperbolic:** Definit canonic de ecuația:
$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 2z$$

2.1 Generatoarele Hiperboloidului cu o Pânză

Pentru hiperboloidul cu o pânză $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$, cele două familii de generatoare pot fi descrise parametric. Un sistem de ecuații parametrice pentru aceste drepte este:

Familia 1 (λ-const):
$$\begin{cases} \frac{x}{a} - \frac{z}{c} = \lambda \left(1 - \frac{y}{b}\right) \\ \frac{x}{a} + \frac{z}{c} = \frac{1}{\lambda} \left(1 + \frac{y}{b}\right) \end{cases}$$
unde λ este un parametru real.

O parametrizare mai directă, utilizată și în cod, pentru un punct (x, y, z) de pe o generatoare în funcție de un parametru unghiular α (care identifică dreapta în familie) și un parametru s (care definește poziția pe dreaptă, similar cu z) este:

- **Familia 1 de generatoare:**
$$\begin{aligned} x(s, \alpha) &= a \left(\cos(\alpha) - \frac{s}{c_{eq}} \sin(\alpha) \right) \\ y(s, \alpha) &= b \left(\sin(\alpha) + \frac{s}{c_{eq}} \cos(\alpha) \right) \\ z(s, \alpha) &= s \end{aligned}$$
- **Familia 2 de generatoare:**
$$\begin{aligned} x(s, \alpha) &= a \left(\cos(\alpha) + \frac{s}{c_{eq}} \sin(\alpha) \right) \\ y(s, \alpha) &= b \left(\sin(\alpha) - \frac{s}{c_{eq}} \cos(\alpha) \right) \\ z(s, \alpha) &= s \end{aligned}$$

Aici, c_{eq} corespunde parametrului c din ecuația canonică a hiperboloidului. Aceste ecuații sunt implementate direct în scriptul Python pentru a genera punctele de pe fiecare linie.

3 Descrierea Implementării Python

Scriptul Python utilizează bibliotecile ‘numpy‘ pentru calcule numerice și ‘matplotlib‘ (cu ‘mplot3d‘ și ‘FuncAnimation‘) pentru crearea graficelor 3D și a animației.

3.1 Parametrii Scriptului

Scriptul începe prin definirea unor parametri cheie:

- `a, b`: Semiaxele elipsei de la $z = 0$ a hiperboloidului.
- `c_eq`: Parametru ce controlează curbura hiperboloidului și, implicit, înclinația generatoarelor. O valoare mai mică accentuează ”gâtul” hiperboloidului.
- `z_max`: Definește lungimea (sau intervalul pe axa z) pentru care sunt desenate segmentele de dreaptă.
- `num_lines`: Numărul de drepte (generatoare) desenate pentru fiecare dintre cele două familii.
- `num_points_per_line`: Numărul de puncte calculate pentru a desena fiecare segment de dreaptă, determinând rezoluția acestora.
- `num_frames`: Numărul total de cadre (imagini individuale) din care va fi compusă animația finală.
- `fps`: Cadre pe secundă (frames per second), determinând viteza animației.
- `initial_elevation`: Unghiul de elevație (în grade) al camerei virtuale pentru vizualizarea 3D.
- `output_filename_gif`: Numele fișierului GIF în care va fi salvată animația.

3.2 Configurarea Scenei 3D

O figură și un subplot 3D sunt create folosind ‘matplotlib‘. Fundalul figurii și al zonei de desenare a axelor este setat explicit pe alb.

```
fig = plt.figure(figsize=(9, 8))
ax = fig.add_subplot(111, projection='3d')
fig.patch.set_facecolor('white')
ax.set_facecolor('white')
```

Listing 1: Configurarea figurii și axelor

3.3 Generarea Liniilor Riglate

Se definesc vectorii `s_values` (coordonatele z de-a lungul unei drepte) și `alpha_values` (parametrii unghiulari pentru distribuirea dreptelor în jurul axei z). Apoi, pentru fiecare familie de linii:

- Se iterează prin valorile `alpha`.
- Pentru fiecare `alpha`, se calculează coordonatele (x, y, z) ale punctelor de pe acea dreaptă folosind ecuațiile parametrice menționate anterior.
- Datele fiecărei linii (coordonate, culoare, grosime) sunt stocate într-o listă `lines_data`. Familia 1 este colorată în albastru, iar familia 2 în roșu.

```
# familia 1
for alpha in alpha_values:
    x_line1 = a * (np.cos(alpha) - (s_values / c_eq) * np.sin(alpha))
    y_line1 = b * (np.sin(alpha) + (s_values / c_eq) * np.cos(alpha))
    z_line1 = s_values
    lines_data.append({'x': x_line1, 'y': y_line1, 'z': z_line1, 'color': 'blue', 'linewidth
        ': 1.2})
```

Listing 2: Generarea datelor pentru o familie de linii

3.4 Funcția de Animație `update_frame`

Această funcție este esențială și este apelată pentru fiecare cadru al animației:

1. `ax.clear()`: Șterge conținutul axelor din cadrul anterior.
2. `ax.set_facecolor('white')`: Reaplică fundalul alb al axelor, deoarece `clear()` poate reseta unele proprietăți.
3. **Redesenarea Liniilor**: Iterează prin `lines_data` și desenează fiecare linie cu proprietățile stocate.
4. **Configurarea Axelor**:
   - `ax.set_xticks([])`, `ax.set_yticks([])`, `ax.set_zticks([])`: Elimină gradațiile numerice de pe axe.
   - `ax.set_xlabel("")`, `ax.set_ylabel("")`, `ax.set_zlabel("")`: Elimină etichetele axelor ("X-axis", etc.).
   - `ax.set_title("Cuadricã dublu-riglatã")`: Setează un titlu pentru grafic.
5. **Setarea Limitelor Axelor**: Limitele axelor (x, y, z) sunt calculate și setate explicit. Acest pas este crucial pentru a menține o vizualizare consistentă și a preveni redimensionarea automată a scenei între cadre, ceea ce ar putea duce la o animație "săltăreață".
6. **Rotirea Vederii**: `ax.view_init(elev=initial_elevation, azim=azimuth_angle_deg)` actualizează unghiul de azimut al camerei, creând efectul de rotație a obiectului în jurul axei verticale.

3.5 Crearea și Salvarea Animației

- `FuncAnimation`: Această clasă din Matplotlib este utilizată pentru a crea obiectul de animație, apelând repetitiv funcția `update_frame` pentru fiecare cadru. Intervalul dintre cadre este determinat de `fps`.
- `PillowWriter`: Animația este salvată într-un fișier GIF folosind `PillowWriter` din biblioteca Pillow (o dependență a Matplotlib pentru scrierea GIF-urilor).

4 Codul Sursă Python Complet

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation, PillowWriter

# --- parametrii hiperboloidului ---
a = 2.0           # semiaxa asociata x pentru elipsa din z=0
b = 1.5           # semiaxa asociata y pentru elipsa din z=0
c_eq = 2.0        # parametrul 'c' din x^2/a^2 + y^2/b^2 - z^2/c_eq^2 = 1
                  # Smaller c_eq -> more pronounced waist/skew.
                  # micsorarea parametrului c_eq modifica inclinatia dreptelor

z_max = 3.0       # lungimea unei drepte
num_lines = 20    # numarul de drepte dintr-o familie
num_points_per_line = 50 # arbitrar, numarul de puncte de pe o dreapta

# --- parametrii animatiei ---
num_frames = 360          # numarul total de frame-uri pentru o rotatie completa
fps = 60                  # cadre pe secunda
initial_elevation = 25    # unghiul de elevatie
output_filename_gif = "animatie_cuadrica_dubluriglata.gif"

# --- setup axe 3D ---
fig = plt.figure(figsize=(9, 8)) # ajustare marime
ax = fig.add_subplot(111, projection='3d')
fig.patch.set_facecolor('white') # fundal alb
ax.set_facecolor('white') # axe de reprezentare albe

```

```
# --- generarea datelor despre linii ---
s_values = np.linspace(-z_max, z_max, num_points_per_line)
alpha_values = np.linspace(0, 2 * np.pi, num_lines, endpoint=False)

lines_data = []

# familia 1
for alpha in alpha_values:
    x_line1 = a * (np.cos(alpha) - (s_values / c_eq) * np.sin(alpha))
    y_line1 = b * (np.sin(alpha) + (s_values / c_eq) * np.cos(alpha))
    z_line1 = s_values
    lines_data.append({'x': x_line1, 'y': y_line1, 'z': z_line1, 'color': 'blue', 'linewidth
        ': 1.2})

# familia 2
for alpha in alpha_values:
    x_line2 = a * (np.cos(alpha) + (s_values / c_eq) * np.sin(alpha))
    y_line2 = b * (np.sin(alpha) - (s_values / c_eq) * np.cos(alpha))
    z_line2 = s_values
    lines_data.append({'x': x_line2, 'y': y_line2, 'z': z_line2, 'color': 'red', 'linewidth'
        : 1.2})

# --- functia de animare ---

def update_frame(frame_number): # functia de desenare a unui singur cadru
    ax.clear() # reimprospatarea cadrului
    ax.set_facecolor('white') # reaplicarea fundalului alb

    for line_props in lines_data: # improspatarea pozitiilor liniilor
        ax.plot(line_props['x'], line_props['y'], line_props['z'],
                color=line_props['color'], linewidth=line_props['linewidth'])

    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_zticks([])
    ax.set_xlabel("")
    ax.set_ylabel("")
    ax.set_zlabel("")
    ax.set_title("")
    ax.grid(False)
    ax.set_title("Cuadrica dublu-riglata")

    # extinderea maxima a liniilor pana la  s = z_max
    # pentru x: a * (cos(alpha) - (z_max/c_eq) * sin(alpha)). valoarea maxima este: a * sqrt
        (1 + (z_max/c_eq)^2)
    max_coord_factor = np.sqrt(1 + (z_max/c_eq)**2)
    axis_limit_x = a * max_coord_factor * 1.1
    axis_limit_y = b * max_coord_factor * 1.1
    axis_limit_z = z_max * 1.1

    ax.set_xlim([-axis_limit_x, axis_limit_x])
    ax.set_ylim([-axis_limit_y, axis_limit_y])
    ax.set_zlim([-axis_limit_z, axis_limit_z])
    ax.grid(True) # This will make the grid visible

    azimuth_angle_deg = frame_number * (360 / num_frames)
    ax.view_init(elev=initial_elevation, azim=azimuth_angle_deg)

    return fig,

print("generarea cadrelor animatiei")
animation = FuncAnimation(fig, update_frame, frames=num_frames,
                          interval=(1000 / fps), blit=False)

try:
    print(f"salvarea animatiei ca: {output_filename_gif}")
    writer = PillowWriter(fps=fps)
    animation.save(output_filename_gif, writer=writer)
    print(f"animatie salvata ca: {output_filename_gif}")
except Exception as e:
```

```
    print(f"eroare salvare: {e}")

print("script terminat")
```

Listing 3: Script Python pentru animația cuadricei dublu riglate

5 Concluzii

Scriptul Python prezentat generează o animație a unui hiperboloid cu o pânză, ilustrând clar cele două familii de generatoare care formează suprafața sa. Prin ajustarea parametrilor, se pot explora diverse forme și configurații ale acestei cuadrice dublu riglate.