5 Serpentis-Реальные данные

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

# === ЧАСТЬ 1: Загрузка и обработка реальных данных ===

real\_data = pd.read\_csv(r"AP14161893.csv.csv")

real\_data = real\_data.rename(columns={'hjd': 'Time', 'mag': 'Magnitude', 'mag\_err': 'Error'})

real\_data = real\_data.dropna()

real\_data = real\_data[real\_data['Magnitude'] < 30]

# Переводим звездные величины в поток

mag0 = real\_data['Magnitude'].min()

real\_data['Flux'] = 10 \*\* (-0.4 \* (real\_data['Magnitude'] - mag0))

# Фазирование

P\_days = 3.371\*365

real\_data['Phase'] = (real\_data['Time'] % P\_days) / P\_days

# Диапазон для масштабирования модели

real\_flux\_min = real\_data['Flux'].min()

real\_flux\_max = real\_data['Flux'].max()

# === ЧАСТЬ 2: Синтетическая модель ===

G = 6.67430e-11

M\_sun = 1.98847e30

R\_sun = 6.957e8

m1 = 0.241 \* M\_sun

m2 = 0.212 \* M\_sun

R1 = 0.254 \* R\_sun

R2 = 0.231 \* R\_sun

P = P\_days \* 24 \* 3600 # в секундах

# ✅ Правильный расчёт большой полуоси

a = (G \* (m1 + m2) \* P\*\*2 / (4 \* np.pi\*\*2))\*\*(1/3)

# Начальные координаты и скорости

r1\_0 = np.array([-m2 / (m1 + m2) \* a, 0])

r2\_0 = np.array([m1 / (m1 + m2) \* a, 0])

v1\_0 = np.array([0, -np.sqrt(G \* m2\*\*2 / (a \* (m1 + m2)))])

v2\_0 = np.array([0, np.sqrt(G \* m1\*\*2 / (a \* (m1 + m2)))])

y0 = np.concatenate([r1\_0, v1\_0, r2\_0, v2\_0])

def derivatives(t, y):

r1, v1 = y[0:2], y[2:4]

r2, v2 = y[4:6], y[6:8]

r = r2 - r1

dist = np.linalg.norm(r)

a1 = G \* m2 \* r / dist\*\*3

a2 = -G \* m1 \* r / dist\*\*3

return np.concatenate([v1, a1, v2, a2])

# Интегрируем на 3 периода

T = 3 \* P

t\_eval = np.linspace(0, T, 2000)

sol = solve\_ivp(derivatives, (0, T), y0, t\_eval=t\_eval, rtol=1e-9, atol=1e-9)

r1 = sol.y[0:2]

r2 = sol.y[4:6]

# === Вычисление синтетической яркости ===

brightness\_raw = []

for i in range(len(t\_eval)):

x1, y1 = r1[:, i]

x2, y2 = r2[:, i]

dx = abs(x1 - x2)

overlap = dx < (R1 + R2)

if overlap:

# Элементарная модель затмения

if y1 > y2:

total\_brightness = 1.0 # малая звезда перекрывает часть большей

else:

total\_brightness = 0.9 # большая звезда перекрывает малую

else:

total\_brightness = 1.9 # сумма потоков двух звезд

brightness\_raw.append(total\_brightness)

brightness\_raw = np.array(brightness\_raw)

# === Масштабирование модели под реальные данные ===

model\_min = brightness\_raw.min()

model\_max = brightness\_raw.max()

brightness\_scaled = real\_flux\_min + (brightness\_raw - model\_min) \* (real\_flux\_max - real\_flux\_min) / (model\_max - model\_min)

# === Построение фаз синтетической модели ===

synthetic\_phase = (t\_eval / (3600 \* 24) % P\_days) / P\_days

# === ЧАСТЬ 3: Построение графика ===

plt.figure(figsize=(10, 5))

plt.scatter(real\_data['Phase'], real\_data['Flux'], color='royalblue', s=10, alpha=0.6, label='Реальные данные')

plt.plot(synthetic\_phase, brightness\_scaled, color='orange', linewidth=2, label='Синтетическая модель')

plt.title("Фазовая кривая блеска 5 Змеи: модель + реальные данные")

plt.xlabel("Фаза (0–1)")

plt.ylabel("Нормированная яркость")

plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.show()