5 Serpentis-затмение

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

# Константы

G = 6.67430e-11 # гравитационная постоянная, м^3 / (кг \* с^2)

# Массы звезд системы 5 Змеи

m1 = 1.8 \* 1.989e30 # масса звезды A, кг

m2 = 1.3 \* 1.989e30 # масса звезды B, кг

# Радиусы звёзд для моделирования затмений

R1 = 2.0 \* 6.957e8 # радиус звезды A, м

R2 = 1.5 \* 6.957e8 # радиус звезды B, м

# === Начальные условия ===

# Положение (x, y) и скорость (vx, vy)

# Оценочный начальный радиус орбиты (приближённо, чтобы были замкнутые орбиты)

r\_orbit = 3.5e11 # м, примерная величина

# Орбитальная скорость по формуле Кеплера

v\_orbit = np.sqrt(G \* (m1 + m2) / r\_orbit)

# Положение звёзд

r1 = np.array([-m2 / (m1 + m2) \* r\_orbit, 0]) # звезда A

r2 = np.array([m1 / (m1 + m2) \* r\_orbit, 0]) # звезда B

# Скорости звёзд (перпендикулярно радиусу для круговой орбиты)

v1 = np.array([0, -m2 / (m1 + m2) \* v\_orbit]) # звезда A

v2 = np.array([0, m1 / (m1 + m2) \* v\_orbit]) # звезда B

# Начальный вектор состояния

y0 = np.concatenate((r1, v1, r2, v2))

# === Система уравнений движения ===

def two\_body(t, y):

r1 = y[0:2]

v1 = y[2:4]

r2 = y[4:6]

v2 = y[6:8]

r = r2 - r1

norm\_r = np.linalg.norm(r)

a1 = G \* m2 \* r / norm\_r\*\*3

a2 = -G \* m1 \* r / norm\_r\*\*3

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

# === Интегрирование ===

# Временной интервал

t\_span = (0, 1.0e8) # увеличим время до ~3 лет, чтобы увидеть несколько оборотов

t\_eval = np.linspace(t\_span[0], t\_span[1], 5000)

# Решение системы

sol = solve\_ivp(two\_body, t\_span, y0, t\_eval=t\_eval, rtol=1e-8)

# Извлекаем координаты

x1, y1 = sol.y[0], sol.y[1]

x2, y2 = sol.y[4], sol.y[5]

# === График орбит ===

plt.figure(figsize=(8, 6))

plt.plot(x1, y1, label='Звезда A (5 Сер А')

plt.plot(x2, y2, label='Звезда B (5 Сер B')

plt.scatter([x1[0], x2[0]], [y1[0], y2[0]], color='red', marker='o', label='Начальное положение')

plt.xlabel('x [м]')

plt.ylabel('y [м]')

plt.legend()

plt.title('Орбиты системы 5 Serpentis')

plt.grid()

plt.axis('equal')

plt.show()

import numpy as np

import matplotlib.pyplot as plt

# Параметры радиусов звёзд (мы их уже знаем!)

R1 = 7.5 \* 6.957e8 # радиус звезды A, м

R2 = 5.1 \* 6.957e8 # радиус звезды B, м

# Светимости звёзд, пропорциональные массе в степени ~3.5

L1 = m1\*\*3.5

L2 = m2\*\*3.5

# Массив яркости

brightness = []

# Проходим по каждому моменту времени

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

# Координаты звёзд

x\_star1 = x1[i]

x\_star2 = x2[i]

# Проверка перекрытия по оси наблюдения (вдоль оси Y)

distance = np.abs(x\_star1 - x\_star2)

if distance < (R1 + R2):

# Есть перекрытие!

# Определяем, кто ближе к наблюдателю по оси наблюдения (вдоль Y)

if y1[i] > y2[i]:

# Звезда 1 ближе к наблюдателю

visible\_luminosity = L1 + max(0, L2 \* (1 - (R1 / R2)\*\*2))

else:

# Звезда 2 ближе к наблюдателю

visible\_luminosity = L2 + max(0, L1 \* (1 - (R2 / R1)\*\*2))

else:

# Перекрытия нет, обе звезды видны полностью

visible\_luminosity = L1 + L2

brightness.append(visible\_luminosity)

# Нормализация яркости

brightness = brightness / np.max(brightness)

# Построим синтетическую кривую блеска

plt.figure(figsize=(8, 4))

plt.plot(t\_eval / (60\*60\*24), brightness, color='orange')

plt.title('Синтетическая кривая блеска системы 5 Serpentis')

plt.xlabel('Время (дни)')

plt.ylabel('Относительная яркость')

plt.grid()

plt.show()